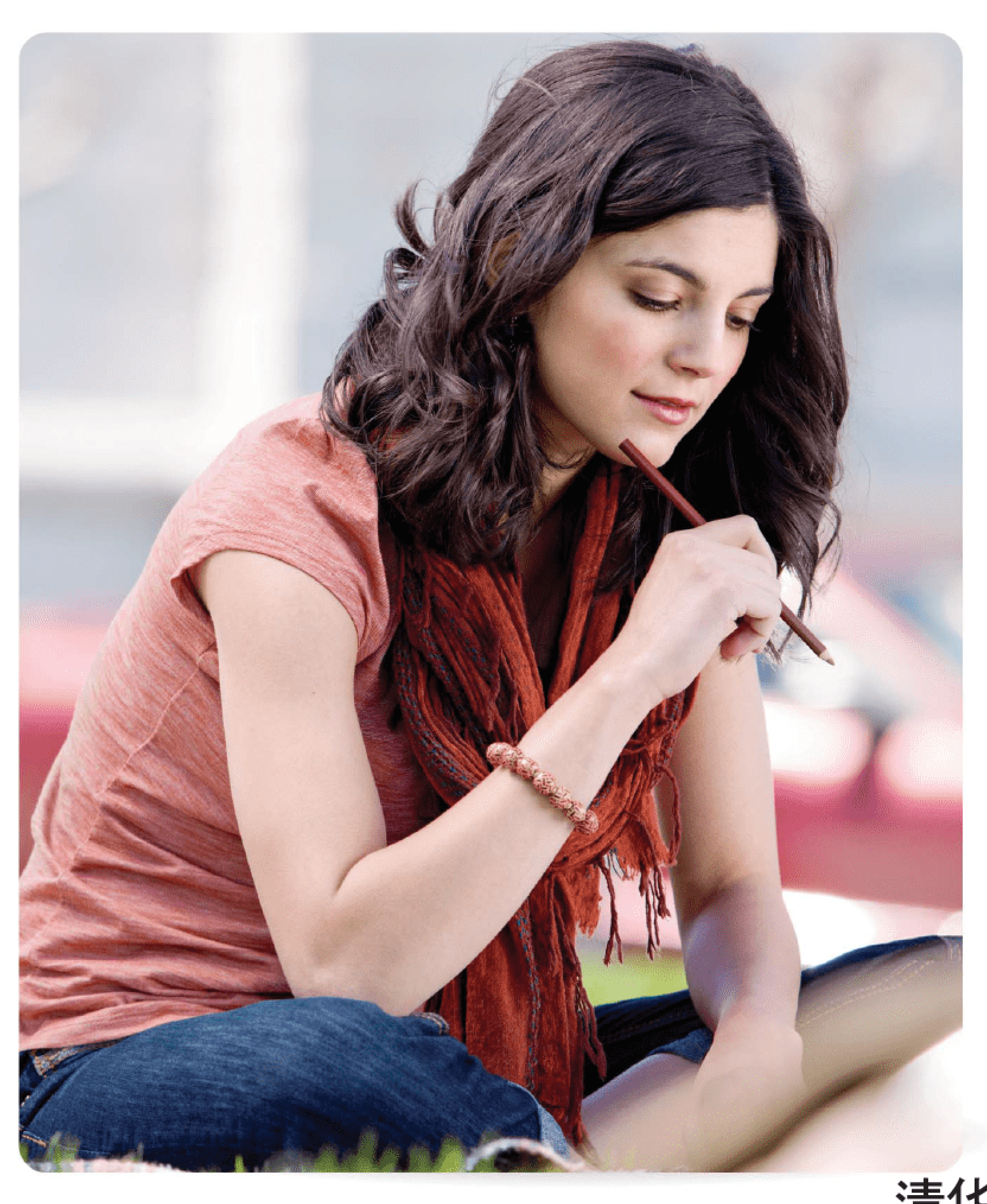



# 软件安全测试及工具应用

何泾沙　编著



清华大学出版社

惠普国际软件人才高等教育系列丛书

# 软件安全测试及工具应用

何泾沙　编著

清华大学出版社

北　京

## 内 容 简 介

本书由三部分组成，共分为 8 章，前两部分主要介绍相关基础知识，包括软件安全基础知识和软件安全测试基本方法，第三部分重点介绍 HP 安全测试软件的使用，包括动态测试软件 WebInspect 和静态测试软件 HP Fortify 系列的使用。本书内容强调理论结合实际，培养学生学习和掌握软件安全测试的基础识和意识，并学习熟练使用 HP 安全测试软件和相关工具。

本书从丰富的理论知识到实际的操作引导，使学生能够全面掌握软件安全测试知识并获得一定的实践经验。本书主要用于软件安全测试及工具应用课程，该课程是一门讲述软件安全基础知识、测试方法以及安全测试软件实际应用的软件工程核心课程，在软件测试专业人才培养体系中占有重要的地位及核心作用。

本书可作为高等院校计算机科学与技术或软件工程等专业本科生相关课程的教材，也可供从事软件测试的工程技术人员和科技工作者参考。



**图书在版编目(CIP)数据**
软件安全测试及工具应用 / 何泾沙 编著. —北京：清华大学出版社，2014.12
(惠普国际软件人才高等教育系列丛书)
ISBN 978-7-302-38490-8

Ⅰ. ①软…　Ⅱ. ①何…　Ⅲ. ①软件可靠性—测试　Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2014)第 260607 号

**责任编辑：**王　军　韩宏志
**装帧设计：**思创景点
**责任校对：**曹　阳
**责任印制：**杨　艳

**出版发行：**清华大学出版社
**网　　址：**http://www.tup.com.cn，http://www.wqbook.com
**地　　址：**北京清华大学学研大厦 A 座　**邮　　编：**100084
**社 总 机：**010-62770175　**邮　　购：**010-62786544
**投稿与读者服务：**010-62776969，c-service@tup.tsinghua.edu.cn
**质 量 反 馈：**010-62772015，zhiliang@tup.tsinghua.edu.cn
**印 装 者：**北京嘉实印刷有限公司
**经　　销：**全国新华书店
**开　　本：**185mm×260mm　**印　　张：**24.25　**字　　数：**605 千字
**版　　次：**2014 年 12 月第 1 版　**印　　次：**2014 年 12 月第 1 次印刷
**印　　数：**1～1700
**定　　价：**55.00 元

产品编号：061437-01

# 前　言

随着信息化系统建设复杂度的提高、网络与数据库的迅速发展及其应用的不断广泛和深入，社会对信息和信息技术依赖性的不断增强，软件的安全性显得越来越重要。不同的软件包含各类信息，有的包含行业机密，有的包含个人隐私信息，有的甚至包含国家机密信息，软件的安全性已经关系到国家政治、军事、经济等各个方面乃至全社会的安全，针对软件安全性进行测试也就显得十分重要。尽管“软件安全”的概念已经为大多数业内人士所熟知，但国内在软件安全测试领域仍存在很大差距，培养软件安全测试方面的人才已经成为高等学校的一项重要任务，这也是编写本书的目的。

本书是针对高等院校计算机、软件工程等专业本科生的教学特点和教学要求，更加强调理论和工程技术应用相结合而编写的教材。本书以介绍软件安全的基础理论知识、软件安全设计与编程的基础知识、软件安全技术的基本知识为背景，以代码安全静态分析和软件安全动态渗透测试知识为支撑，重点是通过介绍 WebInspect 和 HP Fortify 这两款惠普公司开发的软件安全测试工具，使读者能够系统地了解并掌握软件安全测试的理论与实际方法，达到理论与实践相结合的教学效果。

本书由 3 部分构成。第 1 部分包括第 1～3 章，主要介绍软件安全基本概念、安全的软件开发生命周期、软件安全编程以及软件安全相关的技术，为第 2 部分介绍软件安全测试方法进行必要的准备和铺垫。第 2 部分包括第 4 和第 5 章，完整和系统地介绍代码安全静态分析方法和软件安全动态渗透测试方法。第 3 部分包括第 6～8 章，详细介绍惠普公司开发的安全测试软件 WebInspect 和 HP Fortify 的使用方法。

本书旨在为读者提供软件安全测试相关的基础理论和相关软件的使用技巧，并重点讨论软件安全问题的现状和相关的国际标准，介绍软件安全技术中的加密技术、身份验证技术、访问控制技术、安全保障和网络安全的相关知识。全书涉及的内容十分广泛，在作为教材使用时可根据课程要求和学时数在内容、重点和深度方面进行适当的取舍。

本书由北京工业大学何泾沙教授担任主编，参编人员包括蔡建平、轩兴刚、白鑫、杜江浩、郭军、周博、赵斌，以下人员在编写过程中多次参与了研讨活动以及最后的校稿：张玉强、张伊璇、刘公政、陈奕洲、潘力斌、万雪姣、徐琛、张博雍、张跃骞、郑伟。在此向所有参编和参校人员表示衷心感谢。

本书在编写过程中参考了国内外许多相关领域的文献和书籍，也从其他同行的工作中得到了很多启发，在此一并向相关人士表示衷心感谢。

由于作者水平和编写时间所限，书中难免存在各种错误与不足，敬请读者和同行专家批评指正。

编　者

# 目　录

# 第1章　软件安全基础

## 本章导读

本章首先介绍软件安全相关的基本概念、软件安全知识体系和常用名称以及软件安全相关的常用工具。然后介绍软件安全问题及其产生的原因，列举常见的软件缺陷和漏洞，并讲解针对软件安全的测试方法。最后阐述软件安全的相关标准，包括安全规则与规章、软件安全原则和相关国际标准。

应掌握的知识要点：

- 软件安全基本概念；
- 软件安全常用名称；
- 软件安全调试工具；
- 软件的安全问题；
- 软件缺陷和漏洞；
- 针对软件安全问题的测试；
- 软件安全的原则。

## 1.1　软件安全基本概念

当代社会，计算机应用的迅速发展给人们的生活带来极大的便利。软件是计算机应用的重要组成部分，软件的安全性一直是软件开发中的一个关键问题，开发安全性高的软件是软件开发人员追求的目标。开发人员如何在开发的全过程中从根本上提高软件的安全性，是每个软件人员(包括系统分析人员、项目经理、编码人员、测试人员等)必须思考的问题。本节将介绍软件安全相关的基本概念。

### 1.1.1　软件安全概述

20 世纪 80 年代初，IEEE 给出了软件的严格定义：计算机程序、方法、规则和相关的文档资料以及在计算机上运行时所需的数据。软件由三个部分组成，分别是程序、数据和文档资料，通过公式展示为：软件=程序+数据+文档资料。其中，程序是完成特定功能和满足性能要求的指令序列；数据是程序运行的基础和操作的对象；文档资料是与程序开发、维护和使用有关的图文资料。

对软件的类型进行划分，根据不同类型的工程对象采用不同的安全方法很有价值，因此有必要从不同角度讨论计算机软件的分类情况。

按照功能进行划分，软件可以分为系统软件、应用软件和支撑软件三个类别。

系统软件是计算机正常工作的不可或缺的组成部分，它和计算机的硬件紧密配合，控制并且协调计算机系统中的各个组件、相关的软件和数据资源高效地工作。常见系统软件包括操作系统、设备驱动程序以及通信处理程序等。

应用软件是指在专一的领域内开发，为某个特定目标服务的软件。目前，人们在工作和生活中严重依赖计算机，很多应用领域都需要专门的软件支持，在种类繁多的应用软件中，商业数据处理软件所占的比例最大。此外，还有系统仿真软件、工程与科学计算软件、人工智能软件以及各类办公自动化软件和信息处理软件等。

支撑软件是协助用户开发软件的辅助性工具，包括帮助技术人员进行开发的工具和帮助管理人员控制开发进度的工具，如需求原型制作工具、设计工具、编码工具和软件测试工具等。

按照软件的规模进行划分，可分为微型、小型、中型、大型、极大型 6 种。其中，微型是指参加人员数目为 1 人，研制期限为 1 至 4 周，产品的规模，即源程序行数大致为 500 行。小型是指参加人员数目为 1 人，研制期限为 1 至 6 个月，产品的源程序行数为 1000 行～2000 行。中型是指参加人员数目为 2～5 人，研制期限为 1～2 年，产品的源程序行数为 5000 行～50 000 行。大型是指参加人员数目为 5～25 人，研制期限为 2～3 年，产品的源程序行数为 5 万行 10 万行。大型是指参加人员数目为 100～1000 人，研制期限为 4～5 年，产品的源程序行数为 100 万行以上。极大型是指参加人员数目为 2000～5000 人，研制期限为 5～10 年，产品的源程序行数为 1000 万行以上。

按照软件的工作方式，软件可分为实时处理软件、交互软件、批处理软件和分时软件。

按照软件的服务对象范围，软件可以分为项目定制软件和面向市场的产品软件。其中项目定制软件是指受到某个特定客户委托，由软件内容提供机构在合同约束下开发出来的软件。产品软件是面向市场需求，由软件内容提供机构开发出来后直接提供给市场，或是为千百个用户服务的软件，例如办公处理软件、财务处理软件和一些常用工具软件等。

根据 ISO 8402 中的定义，安全性指“使伤害或损害的风险限制在可接受的水平内”。因此，软件安全性是软件的一种内在属性。

软件安全(Software Security)就是使软件在受到恶意攻击的情形下依然能够继续正确运行及确保软件在授权范围内被合法使用的思想，即采用系统化、规范化和数量化的方法来指导构建安全的软件。软件安全是一个较新的专业领域，直到 2001 年才出现了软件安全方面的学术资料，这反映出研发人员、软件架构人员、计算机科学家开始系统地考虑到软件安全的构建。但是，软件安全方面的实践准则尚未得到广泛推广和普遍采用。

从风险分析的角度看，软件安全是关于如何理解软件所引起的安全风险以及如何管理这些风险的学科。目前，“使安全成为软件开发的必需部分”已经成为行业界和政府机构认同的观点。美国国土安全部的国家网络安全处专门建立了 BSI 网站，网址为http://buildsecurityin.us-cert.gov/protal。这个网站由美国国家标准技术研究所、国际标准化组织和电气电子工程师协会共同维护。

软件安全工程化的三个支柱是风险管理、软件安全切入点和安全知识。风险管理是一种贯穿软件开发生命周期的战略性方法；软件安全切入点是在软件安全开发生命周期中保障软件安全的一套最佳实际操作方法，这其中包括代码的审核、体系结构风险分析、基于风险的安全测试、渗透测试、滥用案例、安全需求和安全操作；安全知识是指软件安全知识体系中

的描述性知识、诊断性知识和历史性知识。

软件缺陷是软件安全的关键问题。软件缺陷包括软件实现中的错误异常，如缓冲区溢出；还包括设计中的错误异常，如不成熟的错误处理。随着软件技术的不断发展，软件系统日趋复杂，潜在的安全威胁不断增多，软件中的安全漏洞正在逐年增长。基于互联网的应用软件由于其特点成为风险最高的软件。卡巴斯基病毒实验室给出了恶意软件预报：软件攻击方式会从 Web 转移到文件共享网络。软件攻击演变过程中经历了 Email 攻击、互联网攻击、Web 站点攻击、Torrent 站点分发恶意文件攻击。据统计，当前互联网上攻击次数最多的恶意程序是以下三种程序，分别是：HEUR:Trojan.Script.Iframer，攻击次数为 9 858 304，占攻击的 13.39%；Trojan:Downloader.JS.Gumblar.x，攻击次数为 2 940 448，占攻击的 3.99%；not-a-virus:AdWare.Win32.Boran.z，攻击次数为 2 875 110，占攻击的 3.91%。

受害机器的本地感染数量是特别重要的数据，因为它提供了对威胁数量的统计，这些数据的统计是通过带菌体(而不是通过 Web、Email、网络端口)成功渗透到受害机器的威胁数量。卡巴斯基病毒实验室报告了以下几种最常见的本地感染威胁：HEUR:Trojan. Win32.Generic，检测出该对象的计算机数量为 3 050 753；Net-Worm.Win32. Kido.ih，检测出该对象的计算机数量为 2 924 062；Virus.Win32.Sality.aa，检测出该对象的计算机数量为 1 407 976。

软件安全漏洞是最危险的安全问题之一，软件破坏者利用漏洞可以绕开保护机制对计算机发起攻击。Kido 病毒是传播最广泛的病毒之一，它由 Windows 操作系统的一个关键漏洞引发。浏览器是最常见的攻击对象，虽然浏览器本身可能没有漏洞，但如果浏览器的插件或应用中存在漏洞，计算机仍然可能被感染。卡巴斯基病毒实验室报告了以下最常见的漏洞，分别是：Apple QuickTime Multiple Vulnerabilities，该漏洞文件和应用程序的数量为 165 658 505，占计算机所有漏洞文件程序的 8.37%；Microsoft XML Core Services Multiple Vulnerabilities，该漏洞文件和应用程序数量为 8 906 277，占计算机所有漏洞文件程序的 1.93%，Adobe Flash Player Multiple Vulnerabilities，该漏洞文件和应用程序数量为 7 728 963，占计算机所有漏洞文件程序的 1.67%。

根据著名咨询公司 Garterner Rearch 的报告，在对企业网络的攻击中，超过七成来自应用层，而不是网络层或系统层。软件和应用层的漏洞、入侵和入侵尝试的数目在各种环境中都在增加，而基于互联网的软件漏洞很难识别与检测。从 HTTP 请求来看，当发起的攻击利用了攻击包和系统响应来分析 SQL 攻击、认证蛮力攻击、目录游走攻击、Cookie 欺骗攻击、跨站点脚本(Cross-Site Scripting)攻击和逻辑缺陷攻击等攻击方式时，它们和正常的 HTTP 请求几乎没有差别。

很多企业和组织都会定期发布安全报告来总结一定时期内的威胁信息和趋势。以国内的中国信息安全评测中心发布的漏洞通报为例，基本内容主要包括漏洞的数量和增长率方面的统计。根据漏洞的影响范围、利用方式和攻击后果等属性，对漏洞危害进行量化评分，并将漏洞分为 4 个等级，即危急等级(评分为 10 分)、高危等级(评分为 7 至 9.9 分)、中危等级(评分为 4 至 6.9 分)和低危等级(评分为 0 至 3.9 分)。报告中会列出不同等级的漏洞数量，还会列出对我国使用的主流操作系统和常用应用程序影响比较严重的高危漏洞。根据漏洞的类型来划分，数量最多的漏洞来自跨站脚本漏洞，接下来依次是 SQL 注入、缓冲区溢出、权限管理与控制访问、设计错误、代码注入、资源管理错误、信息泄露、数字错误和目录遍历。

通常修补漏洞的方式有两种，一种是安装相应的补丁程序修补漏洞，另一种是通过软件版本的更新来修补漏洞。

### 1.1.2 软件安全知识体系

软件安全的知识体系可分为三大类：描述性知识、诊断性知识和历史知识。这三类共包括 7 种知识，即原则、方针、规则、弱点、攻击程序、攻击模式和历史风险。攻击程序关系到攻击模式、历史风险和弱点，可根据原则制定方针和规则。

描述性知识包括原则、方针和规则。原则和方针是从方法论的高度(高层体系结构角度)进行定义和描述，规则是从程序代码的角度针对性地进行抽象和统一。描述性知识类提供了一些建议，说明需要执行的内容和在构建安全软件时应该避免哪些方面。

诊断性知识包括攻击模式、攻击程序和弱点。诊断性知识不仅包括关于实践的描述性陈述，其更重要的目标是帮助操作人员识别和处理导致安全攻击的常见问题。攻击模式采用较抽象的形式来描述常见的攻击程序，这种形式能应用于跨越多个系统的情形，即在多个系统中均存在的攻击模式，该知识可被安全分析人员所利用，如基于滥用案例的可靠性检测等。攻击程序描述了弱点实例如何用来对特定系统造成特别的安全危害。弱点知识是对真实系统中出现过并且已经报告的软件弱点的描述，较著名的弱点和攻击知识库包括 CVE(Common Vulnerabilities and Exposures)知识库、公共攻击模式列举和分类(Common Attack Pattern Enumeration and Classification)知识库。

历史知识类包括历史风险，有些情况下也包括弱点的历史数据库。这类知识还包括对在实际软件开发中所发现的特定问题的详细描述，以及该问题产生的影响。

描述性知识从战略角度进行描述，主要包括长期积累和提炼出来相对抽象的知识。诊断性知识从战术角度进行描述，可能与具体系统相关，攻击模式从攻击角度描述，弱点从防御角度描述。历史知识库是知识的历史积累和前后关联的总结。

### 1.1.3 软件安全的相关领域

软件安全不是一门孤立的学科，有一些与其紧密联系的相关领域。其中软件工程是软件安全相关领域中最重要的部分。

#### 1. 软件工程

软件工程是一门研究如何使用系统化、规范化、数量化等工程原则和方法进行软件开发和维护的学科。软件工程采用工程的概念、原理、技术和方法来开发维护软件，把成熟的管理方法和先进的软件开发技术结合起来，应用到软件开发和维护过程中来解决软件危机问题，生产出无故障的、及时交付的、不超出预算的和满足用户需求的软件。1993 年，IEEE 给出了软件工程的定义：软件工程师将系统化的、规范的、可度量的方法应用于软件的开发、运行和维护过程，即将工程化应用于软件中方法的研究。

软件工程是一种层次化技术，它的最底层是质量保证层。由最底层至最上层分别为过程层、方法层和工具层。最后这三个层次也称为软件工程的三要素。与任何工程方法一样，软件工程以质量为焦点，以全面质量管理及现代管理理念为软件工程的基础。全面的质量管理和质量需求是推动软件开发过程不断改进的动力，这种改进的动力使得更成熟的软件工程方

法不断涌现。

软件工程的基础是过程层。软件工程过程是为获得软件产品，软件研发人员在软件工具支持下完成的一系列软件工程活动。过程层将方法和工具结合，定义一组关键过程区域的框架，同时定义了方法的使用顺序、必要的文档和管理开发各个阶段完成情况的资料。方法层提供了各种软件开发方法。方法覆盖了一系列任务，包括项目计划与估算、需求与设计、程序与测试和运维方法等。工具层为软件工程方法提供了自动的或半自动的软件支撑环境。使用软件工程工具可以有效地改善软件开发过程，提高软件开发效率，降低开发成本。目前已有许多软件工具，这些工具集成起来成为计算机辅助软件工程的软件开发支撑系统。该支撑系统将各种软件工具、开发机器和用于存放开发过程信息的工程数据库组合起来，形成软件工程环境。

软件工程是一门内容广泛的学科，所依据的基础理论极其丰富，包括数学、计算机科学、经济学、工程学、管理学和心理学等其他学科，其研究的内容包括软件开发技术和软件管理技术。软件开发技术又包括软件开发方法学、软件工具和软件工程环境。软件管理技术包括软件度量、项目估算、进度控制、人员组织、配置管理和项目计划。

### 2. 软件保障

根据美国国家标准与技术研究院的定义：软件保障(Software Assurance，SA)是确保软件过程和产品符合需求、标准和规程的有计划的和系统的行动集合。软件保障可以帮助达到两个目标：可信性，即没有可利用的弱点存在，无论是来自敌意的还是无意的企图；执行可预见性，可证明信任软件在执行时依照所希望的那样进行工作。这些行为包括需求分析、测试、验证以及报告。软件保障涉及两个软件属性：一个属性是质量，也就是说软件保障是软件质量保障的简短形式；另一个属性是可靠性，特别是可靠性中最重要的内容，即安全。

### 3. 软件质量

软件质量是贯穿软件生存期的一个极重要的元素，是软件开发过程中所使用的各种开发技术和验证方法的最终体现。因此，在软件生存期中要特别重视软件质量的保证，以生成高质量的软件产品。IEEE 给出了软件质量的明确定义：系统、部件或过程满足明确需求。也就是说，为满足软件的各项精确定义的功能和性能需求，遵守文档化的开发标准，需要相应地给出或设计一些质量特性及其组合。如果这些质量特性及其组合都能在产品中得到满足，则这个软件产品就是高质量的。软件质量反映了以下三方面的问题：软件需求是度量软件质量的基础，不符合需求的软件就不具备质量；在各种标准中定义了一些开发准则，用来指导软件人员用工程化的方法来开发软件，如果不遵守这些开发准则，软件质量就得不到保证；开发准则中会有一些隐含的需求没有明确地提出来，例如，软件应具备良好的可维护性，如果软件只满足那些明确定义了的需求而没有满足这些隐含的需求，也不能保证软件质量。

以下基本概念从不同的方面体现了软件的质量：

- 软件可靠性：在规定条件及时间内，软件不引起系统失效的概率。它与软件中的缺陷、系统输入和系统使用有关；
- 可维护性：产品在规定的条件和时间内，按规定的程序和方法进行维修，保持或恢复到规定的状态的能力；

- 可用性：产品在任意时刻需要和开始执行任务时，处于可工作或可使用状态的程度；
- 安全性：将伤害或损坏的风险限制在可接受水平内的能力；
- 机密性：避免未经许可泄漏信息的能力；
- 完整性：避免不适当地执行更改的能力；
- 经济性：通常用生命周期费用(Life Cycle Cost)表示软件开发过程中的成本开销。它可以定义为：在产品的生命周期内，在产品设计、研究和研制、投资、使用、维修及保障中发生的或可能发生的一切直接的、间接的、派生的或非派生的费用的总和。

软件质量的模型包含三层：高层软件质量需求评价准则包括了软件的正确性、可靠性、可维护性、效率、安全性、灵活性、互连性和可使用性；中层软件质量设计评价准则包括了可追踪性、完备性、一致性、精确性、简单性、模块独立性、通用性、可扩展性、自检性、执行效率、存储效率、存取审查、操作性、通信性、通信共享性和数据共享性；低层软件质量度量评价标准由使用单位自行制定。

### 4. 软件可靠性

软件可靠性工程是预计、测量和管理以软件为基础的系统的可靠性，最大限度地满足用户要求的应用科学。从软件特性和用户需求分析开始，软件可靠性工程(Software Reliability Engineering)贯穿于软件生命周期的全过程，主要包括以下三方面的内容：

- 为满足用户对软件可靠性的要求，必须给出关于软件可靠性的规范说明。在做软件可靠性设计时，甚至要在其他软件质量指标间做出某些权衡，如可适当增加开发成本、延长测试时间等，以求得更高的软件可靠性。
- 软件可靠性的量测与分析技术。
- 软件工程中保证软件开发的方法和技术。

软件可靠性工程的研究范围主要是：软件可靠性定量评测；软件可靠性的设计与管理；软件可靠性保证技术。

软件可靠性模型是指为预计或估算软件的可靠性所建立的结构和数学模型。建立可靠性模型是为将复杂系统的可靠性逐级分解为简单系统的可靠性，以便定量预计、分配、估算和评价复杂系统的可靠性。一般软件可靠性模型分两大类：软件可靠性结构模型和软件可靠性预计模型。

软件可靠性结构模型：依据系统结构逻辑关系，对系统的可靠性特征及其发展变化规律做出可靠性评价的模型。此模型既可用于软件可靠性综合评价，又可用于软件可靠性分解。

软件可靠性预计模型：用来描述软件失效与软件缺陷的关系，借助这类模型，可对软件的可靠性特征做出定量的预计和评估。依据软件缺陷与运行剖面数据，利用统计学原理建立二者之间的数学关系，获取开发过程中可靠性变化、软件在预定工作时间的可靠度、软件在任意时刻发送失效的平均值，以及软件在规定时间间隔内发生失效次数的平均值等数据。这里评估与预计有区别，评估是对现有情况进行评价，预计是依据现有情况及评估结果，对未来可能发生的情况进行科学推断。

软件可靠性通常包括软件安全性的要求，但软件可靠性不能完全取代软件安全性，因为安全性要求包括在非正常条件下不发生安全事故的能力。

### 5. 软件容错性

软件容错性是指软件运行时，能对非正常因素引起的运行错误给出适当的处理或信息提示，使软件正常结束运行。容错技术是软件可靠性技术中一个非常重要的方面。容错技术是对某些无法避开的差错，使其影响减至最小的技术。软件容错技术提供足够的冗余信息与算法程序，使系统在实际运行中能允许、预见、判别、纠正运行中可能出现的错误，恢复和保持系统的正常运行。冗余包括结构冗余、信息冗余和时间冗余。

容错技术包括以下几个主要方面：

- 故障检测技术：故障检测技术是在软件中的故障暴露时，能对由此引起的故障产生响应的过程。在进行故障检测时，需要用到一个重要概念：软件断言。软件在宿主系统中运行时，使其中的诊断程序能对其进程或功能是否正确做出判断的条件称为软件断言。
- 故障恢复技术：软件执行过程中不需要对每一步都进行检测。通常在软件执行过程中设置若干检测点，而恢复点是执行过程中预置的能保存和设定的用于恢复的起始状态点。故障恢复一般有两种策略，即前向恢复和后向恢复。
- 破坏估计：为检测到的故障提供一种有效恢复手段以使系统恢复到正常工作状态或某种预置的状态是容错的主要策略。但是，完整的容错技术除了检测和恢复外，如何对故障引起的破坏做出正确的估计，隔离故障，以及如何在恢复后提供继续服务等技术不可缺少。
- 故障隔离技术。主动地采取措施，防止故障破坏性蔓延的技术。采用了故障隔离技术的系统，由于抑制了故障的扩散，有利于容错的实现。
- 继续服务。系统从故障中恢复到故障前某一状态或预先设计好的其他状态，经历恢复后得到的服务能够被需求规范接受。

### 6. 软件应用安全

应用安全关注应用系统的安全性或网络应用层的安全性，包括如何保护软件以及运行的系统，使用的关键方法包括沙盒(Sand Boxing Code)防御恶意代码的代码混淆、监控程序运行时状态(例如输入操作的监控)、强制使用安全策略等。软件安全关注如何构建安全的软件，包括如何设计安全的软件、如何确认软件的安全性。

网络应用层的应用安全是网络安全的自然延伸，通过一些常见方法(如打补丁、输入过滤等)提供一种反应性(reactive)的安全。简言之，应用安全主要针对发现和修改已知的安全问题，而软件安全是为了安全目标而设计、构建、测试软件的过程。软件安全实践者试图构建积极性(proactive)的能抵御各种攻击的安全软件。虽然在应用安全中，通过观察到达 80 端口的 HTTP 流量来停止缓冲区溢出有实际价值，然而通过软件安全的方法，设计或修改代码来完全避免缓冲区溢出则更加完美。

根据应用系统的不同，可将应用安全进一步细分，例如数据库安全、Web 安全、网络游戏安全和电子邮件安全等。与应用安全处于同一层次的安全性问题包括：操作系统安全，主要研究操作系统软件的安全性；网络安全，研究网络系统的安全性。

### 1.1.4 软件安全常用名称及定义

- 内存：计算机系统上被分配用于暂时存放 CPU 中的运算数据，以及与硬盘等外部存储器交换的数据。
- 寄存器：处理器用于暂存信息的地方。所有处理器都在寄存器上完成操作。
- X86：X86 是一组通常与 Intel 相关的计算机架构。X86 架构是小字节序系统。
- API：应用编程接口是包含特殊函数功能的程序组件，这些函数功能可供程序员在程序中使用。
- 汇编代码：汇编语言是含有一些简单操作的低级程序语言。对汇编代码进行汇编将得到机器代码。在 C/C++代码中编写内联汇编程序通常会产生更有效快捷的应用程序。但这种代码很难维护，可读性不好，代码通常较长。
- 大字节序：在大字节序系统中，最重要的位的字节首先被存储。可扩充处理器架构使用了大字节序架构。
- 小字节序：在小字节序系统中，先存储不重要的字节。X86 架构使用了小字节序架构。
- 缓冲区：已经分配的一段大小确定的内存空间。当数据在两个运行速度或作业量不同的设备间传递时，通常使用缓冲区作为数据的临时存储区。使用 malloc 函数在堆上分配动态的缓冲区，而定义静态变量时，缓冲区被分配到堆栈上。
- 字节码：字节码是介于人能理解的高级语言代码和计算机能理解的机器代码之间的程序代码。对于诸如 Java 的语言(独立于平台的语言)，字节码是很有用的中间码。每个系统的字节码解释程序解释字节码的速度可能比完全解释高级语言更快。
- 编译器：将程序语言转换成可执行的机器码的工具。
- 解释器：阅读并执行程序代码的工具。编译器将代码编译成机器码并存储。不同于编译器，解释器只是阅读高级源代码。解释器的优势是独立于平台，程序员不需要在各种平台上编译源代码，每个含有语言解释器的系统都可以运行相同的程序代码。
- 调试器：调试应用程序时的运行环境或是类似于运行程序的虚拟机的软件工具。该软件允许技术人员在调试应用程序时修改问题。调试器允许终端用户修改环境，如应用程序依赖的内存。
- 反汇编程序：将机器代码程序转变为汇编代码的软件工具。流行的反汇编程序包括 Objdump(通常包含在 UNIX/Linux 系统中)和更强大的工具 IDA Pro。
- DLL：动态链接库文件都有.dll 扩展名。DLL 是运行在 Win32 系统上的程序组件，包含可被其他应用程序使用的功能。
- 堆：堆是应用程序使用的一块内存区域，在运行应用程序时动态分配。静态变量和使用 malloc 函数接口分配的数据一起存储在堆栈上。
- 机器语言：机器代码是可被处理器理解和执行的代码。编译器将程序员用高级语言设计的代码转换成机器语言。
- malloc：调用 malloc 函数将在堆中动态分配 n 个字节。许多漏洞都与这种内存分配方式有关。
- memset/memcpy：调用 memset 函数将用确定字节的特定字符来填充堆缓冲区。调用 memcpy 函数将堆上的缓冲区内的确定数目的字节复制到内存的缓冲区内。这个功能

类似于 strcpy 函数的功能。

- sandbox：一种用来控制代码执行的结构。在 sandbox 中执行的代码不会影响外部系统。用户需要运行可变代码(如 Java applet)时，这种结构对安全很有用。
- shellcode：传统上，shellcode 是执行 shell 的代码。现在，shellcode 有了更广泛的含义，通常定义成功 exploit 所执行的代码。
- 软件缺陷：不是所有的缺陷都是漏洞，如果不能被利用，软件缺陷就不是漏洞。软件缺陷可以简单到软件的界面组件没有对齐等。
- 堆栈：堆栈是用来存储临时数据的内存空间，在程序运行期间不停地伸缩。一般的缓冲区溢出就发生在内存的堆栈空间。当缓冲区溢出时，溢出的内容将覆盖被保存的返回地址，从而使恶意用户获得对程序的控制权。
- 虚拟机：可执行代码的平台的模拟软件。虚拟机允许代码执行，而不需要针对特定的硬件处理器对代码进行修改。这使得代码具有可移植性且对平台具有独立性。
- strcpy/strncpy：strcpy 是 Libc 函数，其调用经常被误用，因为它把数据从一个缓冲区复制到另一个缓冲区而没有对数据大小进行限制。如果源缓冲区是用户的输入，则可能发生缓冲区溢出。strncpy 也是 Libc 函数，该函数调用相当于 strcpy 调用增加了一个表示数据大小的参数；但是，如果数据不是正确地动态产生或没有计算尾部的空字符，这个表示数据大小的参数也会算错。
- Exploit：典型的 Exploit 是一个程序，通过使用该程序可以触发一个软件漏洞并为攻击者所利用。
- 0 day：也被称为 Zero Day。0 day 意味着一个漏洞在被公开发布前或当时，关于它的利用(exploit)已经被发布或使用。
- 缓冲区溢出：当向一个已经分配了确定存储空间的缓冲区内复制多于该缓冲区处理能力的数据时，将发生缓冲区溢出，溢出包括堆溢出和堆栈溢出。
- 格式化字符串缺陷：格式化字符通常在参数数量可变的函数中使用，例如 printf、fprintf、syslog 等函数。输出数据时，这些格式化字符串拥有相关数据并进行适当的格式化。一些情况下，如果格式化字符串没有被明确定义，而用户又可以给函数输入数据，就可能使缓冲区出现格式化字符串缺陷。
- 堆腐烂：堆溢出通常被称为堆腐烂缺陷。因为当堆栈上缓冲区溢出时，数据经常溢出到其他缓冲区；而在堆上，数据则溢出在内存上，不管这些内存是否重要、有用还是可利用。堆腐烂缺陷是发生在内存上堆区域的漏洞，这些缺陷可以多种形式出现，包括 malloc 函数的实现、静态缓冲区的溢出，和堆栈不一样的是只有很多条件都满足时，才可以利用堆腐烂缺陷。
- 堆栈溢出：当堆栈空间的缓冲区过载时，就发生堆栈溢出。发生堆栈溢出时，返回地址将被覆盖，从而允许任意代码的运行。最常见的可利用漏洞就是堆栈溢出。字符串函数(如 strcpy 等)是在源代码中寻找堆栈溢出的出发点。
- 漏洞：漏洞是可被利用的潜在缺陷。大部分有真实意义的漏洞只是特定的软件缺陷。但逻辑错误也是漏洞。例如，缺少口令或允许空口令都是漏洞。而逻辑错误和设计错误并不是软件缺陷。
- Rootkit：可用来获取计算机的未经授权的远程访问权限并发动其他攻击，可能的功

能包括监视击键、更改系统日志、创建后门、发起攻击等。自身隐蔽性好，具有操作系统内核程序的能力，一般针对专门的操作系统类型而设计。

- 分布式拒绝服务(DDoS，Distributed Denial of Service)：DDoS 攻击是一种计算机自动执行的攻击方式，可能使网络服务(如 Web 服务器或文件服务器)超载或停止，其目的是使得特定服务在一定时间内不可用。

## 1.1.5 软件安全调试工具

### 1. 反汇编器

反汇编器可将二进制代码作为输入，生成包含整个或部分程序的汇编语言代码的文本文件的程序。汇编语言代码是目标代码简单的文本映射，这个转换过程较简单。反汇编是一个与处理器相关的过程。IDA Pro 的基本特点是功能全面，能为给定的函数生成有用的流程图。这些流程图基本上都是逻辑图形，显示了反汇编后的代码块，并以一种可视化的方式来展现代码中的每个条件跳转如何影响函数流程。每个盒子代表一个代码片段或函数流程中的小部分。盒子之间用箭头连接，箭头按是否满足条件跳转来显示代码流程。

### 2. 调试器

调试器可以让软件开发人员在程序运行时观察程序的运行过程和状态。它的两个基本功能是设置断点和代码跟踪。断点允许用户选择程序中的任何位置的某个函数或代码行，一旦程序运行到这一行，就告诉调试器暂停程序的运行。当程序运行到断点时，调试器停止程序的运行，并显示程序的当前状态。调试器还允许用户在程序运行时跟踪执行(也叫单步跟踪)。跟踪意味着程序执行一条代码后暂停，允许用户观察(甚至改变)程序状态。然后，可继续执行下一条代码，并不断重复这个过程。

在反汇编模式下使用调试器，能够单步执行反汇编后的代码，观察指令运行时的状态，如寄存器状态和内存转储、活动的堆栈区。调试器可分为两种：用户模式调试器和内核模式调试器。用户模式调试器是较普通的调试器，通常是软件开发人员使用，在用户模式下作为一般的应用程序运行，只能用来调试用户模式的应用。内核模式调试器的功能要强大得多，它们允许无限制地控制目标程序，能对发生在应用程序代码中和操作系统代码中的事件提供完整的视图。

用户模式调试器将自己附着在另一个进程上，并完全控制该进程。其优点是便于安装和使用，但其缺点是仅能观察一个进程，且仅能观察这个进程中的用户模式代码。如果要调试的代码有内核模式的组件，则无法跟踪内核中运行的操作系统代码。

用户模式调试器 OllyDbg 内置了强大的反汇编器，且该反汇编器具有强大的代码分析能力，可以识别循环、switch 控制块以及其他主要的代码结构。它还提供大量不同的视图，包括列出模块中的导入和导出、显示被调试者拥有的窗口和其他对象的列表、显示当前的异常句柄链等。它还包括一个内置的编译和修补引擎，可在程序中的任何区域输入汇编代码，然后将更改后的程序编译运行。

内核模式调试器与用户模式调试器不同，它不是运行在操作系统上的程序，而是与系统的内核处于同等地位的组件。内核模式调试器通常面向内核级开发人员，如设备驱动程序的

开发人员。它的一个主要功能是设置底层断点，即在操作系统底层的代码中设置断点。WinDbg 可进行内核调试，其内核调试模式是在与运行 WinDbg GUI 的系统相互独立的系统上远程执行的，采取的办法是目标系统用 DEBUG 开关项(在配置文件 boot.ini 中设置)引导。被调试的程序与运行 WinDbg 的控制系统通过串行的假调制解调器电缆(null-modem cable)或高速 Fire Wire(IEEE 1394)连接。

### 3. 反编译器

反编译器接收可执行的二进制文件，试图从中生成可读性好的高级语言代码。其思想是逆向编译的过程，以获取最初的源代码文件或接近于最初源代码的文件。

Boomerang 是一种开放源代码的反编译器，主要用于反编译机器代码。反编译二进制代码非常困难，其次是反编译汇编代码，即将汇编代码转变为高级语言代码。一个实例是 Interactive Decompiler，IDC 是一个交互式反编译器，能将汇编语言代码转换成可读性良好的类 C 语言代码。

### 4. 系统监控工具

系统监控工具用来显示操作系统收集到的有关应用程序及其环境的信息。由于程序与外部世界的所有通信都要经过操作系统，因此通常使用操作系统来提取这些信息。系统监控工具可以监控网络活动、文件访问和注册表访问等。有些工具可以展示程序使用操作系统对象的情况，如互斥对象、管道、事件等。Micrsoft Technet 站点提供了如下多个 Windows 系统监控工具：

- FileMon：此工具可以用于监控程序和操作系统之间所有文件系统级的通信，还可以用来显示在系统中运行的每个进程所产生的文件 I/O。使用这个工具能观察到每个打开的文件或目录，以及系统中任何进程执行的读/写操作。
- Process Explorer：此工具可以用于显示进程、加载在各自地址空间中的 DLL 文件、每个进程内的对象句柄、打开的网络连接的详细信息，还可以显示 CPU 以及内存的使用曲线图等信息，此外还可以显示某些与代码相关的细节，如每个进程中各线程的用户栈和内核栈以及符号信息。
- TCPView：此工具可以用于监控每个进程的所有活动的 TCP 和 UDP 网络连接，显示关于某个连接由某个进程打开的列表，以及连接类型、端口号和另一端的系统地址。对显示木马和蠕虫行为很有帮助。
- Autoruns 工具：此工具可以用于显示完整的开机时启动的项目，还可以显示调度任务、系统服务、启动阶段执行的项目、驱动等内容。木马程序通常自动加载到自启动项中，使得系统一开机就运行，所以该工具可用来发现可疑的自启动项目、驱动和系统任务等。

## 1.2　软件安全问题的现状

现代信息生活中，计算机系统的安全问题越来越受到重视。软件是组成计算机应用的重要部分，当软件由于漏洞而遭受攻击，或者运行期间出现错误时，会给用户带来巨大损失：

如犯罪分子利用软件漏洞来获取有价值的信息，用于牟取利益；又如软件因为开发时没有考虑运行时的具体情况，而造成运行的突然崩溃等。

### 1.2.1　当前的软件安全问题

从用户角度出发，典型安全问题表现在以下 4 个方面：

- 使用某些交易软件的过程中，某些敏感信息(如个人身份信息、个人卡号密码等信息)被敌方获取并用于牟利。
- 访问某些网站时，服务器响应缓慢，或者服务器由于访问量造成负载过大，造成突然瘫痪。
- 系统中安装了存在漏洞的软件，漏洞没有解决，攻击者找到漏洞并对该系统进行攻击，造成系统瘫痪。
- 用户耗费精力完成了一幅漂亮的风景画，在互联网发布，没有考虑版权，被他人随意使用却无法问责。

因此，这些安全问题应该在软件开发过程中就充分考虑到。在新时期，对软件的开发提出了两个新要求：使软件更高级和提高可扩展性。这两个要求促进了软件工程应用和研究的发展，但也使软件安全变得更富有挑战性。一方面，软件复杂性提高，安全问题也趋于复杂，无法得到全面考虑，而工程进度又迫使开发者不得不在一定时间内交付产品，代码越多，漏洞和缺陷也就越多；另一方面，软件的可扩展性要求越来越高，系统升级和性能扩展成为很多软件必备的功能；可扩展性好的系统，由于能够用较少的成本扩充功能，受到开发者和用户的欢迎；但针对可扩展性必须进行相应的设计，软件结构变得复杂。添加新的功能的同时，也引入了新的风险。

怎样消除这些安全问题？首先，大多数人可以想到的方法是软件测试，通过测试来减少软件中缺陷或漏洞的数量。但是，由于软件系统规模越来越大，软件开发的进度要求越来越高，不可能在有限时间内考虑安全方面的所有问题，即使进行了全方位软件测试，也只能覆盖所有测试案例中的很小一部分。由于工程进度问题，实际上在测试时不可能兼顾全面，往往只是采用了一些具有代表性的测试案例来进行测试，但这些测试案例在设计时又不能保证具有最全面的代表性。如果要将所有问题考虑周全，除非进行穷举测试，而完成这种穷举测试目前是不现实的。

因此，软件测试无法完全保证软件的安全性。一方面是想要实现全面的测试，找出全部的错误，另一方面又要保证工程的进度，解决用户的问题，两者存在矛盾，只能在其中找到平衡点。

关于测试的另一个问题是全面测试。一般情况下是针对所有可能出现的隐患进行测试，但这需要对软件的隐患具有全方位的预见性。而有些情况下，很多隐患是在运行期间才显露出来的，软件开发者很难在开发阶段预见到所有可能出现的隐患，容易让测试陷入盲目。因此，测试只能减少软件安全问题的发生，但是不能完全解决安全问题。业界大都公认一个事实：几乎所有的已发布软件都存在安全隐患。这里可以得出如下结论：任何软件都是不安全的，包括进行了广泛测试的知名接口、协议(如 TCP/IP)等，很多情况下都会成为攻击的目标。另外，从技术方面来讲，软件的安全问题(如缓冲区溢出、异常处理不当、线程同步问题没有考虑等)是普遍存在的，无法完全避免，黑客可采取多种手段入侵。以网络软件为例，黑客可

能通过互联网未经授权获得信息，或者利用软件缺陷来控制用户系统并展开攻击。随着网络应用的更趋丰富，用户对网络服务的依赖也相应增加(如网上银行、网上股票、网上游戏等)，这也导致了入侵方法的多样化和复杂化，使得安全问题更加明显。

另一个解决安全问题的方法是在测试前减少安全隐患。在设计、编码阶段，熟练的软件设计人员和软件工程师完全可以尽可能多地将安全问题进行考虑并加以解决。如果在程序设计时就能尽力全面考虑安全问题，对软件的安全性就会有更好的保证，可以大大减轻测试负担。

### 1.2.2　造成软件不安全的原因

通常，软件不安全性的受害者就是其直接用户。从用户角度来看，软件的不安全性主要体现在两个方面。

一方面软件在运行过程中不稳定，出现异常现象、得不到正常结果，或在特殊情况下由于一些原因造成系统崩溃。比如：由于异常处理不当，软件运行期间遇到突发问题，处理异常之后无法释放资源，导致这些资源被锁定无法使用；由于线程处理不当，软件运行中莫名其妙得不到正常结果；由于网络连接处理不当，网络软件运行过程中，内存消耗越来越大，系统越来越慢，最后崩溃；由于编程没有进行优化，程序运行消耗资源过大等。

另一方面，敌方利用各种方式攻击软件，达到窃取信息、破坏系统等目的。比如：敌方通过一些手段获取数据库中的明文密码；敌方利用软件的缓冲区溢出，运行敏感的函数；敌方利用软件对数据的校验不全面，给用户发送虚假信息；敌方对用户进行拒绝服务攻击等。

通常情况下，大多数安全问题在软件运行过程中发生，而负责软件系统运行的技术管理人员或软件的个人用户，并不是专业软件开发人员。此时他们往往无法给出直接的应对方案，虽然可以依靠一些简单方法，如优化操作系统、优化网络、优化数据库管理系统或设置额外的操作权限来应对这些剧增的安全问题，但实际上，这些方法不能根除问题。此时，软件的生产单位就需要投入大量成本来进行软件的维护。

软件出现安全隐患，并造成损失，一方面是由于攻击者的猖獗，但从开发者角度，几乎都有一个共同的基本原因，那就是由于在设计、编码、测试和运行阶段，没有发现软件中的各种漏洞，导致软件的不安全。

从严格定义上讲，软件安全隐患一般可分为两类：错误和缺陷。错误是指软件实现过程出现的问题，大多数错误可以很容易发现并修复，如缓冲区溢出、死锁、不安全的系统调用、不完整的输入检测机制和不完善的数据保护措施等；缺陷是一个更深层的问题，它往往产生于设计阶段并在代码中实例化且难以发现，如设计期间的功能划分问题等，这种问题带来的危害更大，但不属于编程范畴。

下面阐述软件不安全的原因。首先，站在软件开发者的角度，软件不安全的原因可以归纳为以下几种：

1) 软件生产没有严格遵守软件工程流程。由于缺乏经验或主观(如片面追求进度)原因，软件的设计者和开发者没有进行统一管理，可以在软件开发周期的任意时候，随意删除、新增或修改软件需求规格说明书、威胁模型、设计文档、源代码、整合框架、测试用例和测试结果、安装配置说明书，使得软件的安全性减弱。

大多数系统软件或其他商业软件的结构都庞大繁杂，而且由于考虑到软件的扩展性，它

们的设计更加巧妙，复杂性可能更高。在运行过程中，这些系统又可以在大量不同的状态之间转换，这个特性增加了持续安全运行的难度。面对不可避免的安全威胁和风险，项目经理和软件工程师必须从开发流程做起，让安全性贯穿整个软件开发周期。就大多数相对成功的软件工程案例而言，如果项目经理和软件工程师针对软件缺陷进行系统的训练，可以避免软件的许多安全缺陷。

2) 编码者没有采用科学的编码方法。在软件开发过程中没有考虑软件可能出现的问题，仅将能想到的问题停留在实验室内进行解决。实际上，有些程序，在实验室阶段根本不会出现安全隐患，如：

```
void function(char *input)
{
    char buffer[16];
    strcpy(buffer，input);
}
```

表示将input字符串复制到 buffer 中，如果没有考虑缓冲区溢出，即使在开发阶段的测试过程中让这个函数产生缓冲区溢出，也不会产生攻击效果。只有在精心设计后，才可能对系统造成攻击。因此在开发阶段很难意识到这个问题，使得软件留下安全隐患。

3) 测试不完全。主要是测试用例的设计无法涵盖所有安全问题。如常见的通过用户名和密码进行登录的表单，一般测试用例只是设计输入正确的用户名和密码，看能否正常登录；再输入错误的用户名和密码，看能否得到相应的错误提示。但攻击者如果输入某些与 SQL 注入有关的值，就可能在不需要知道用户名和密码的情况下登录到系统，甚至知道系统中的其他信息或对系统中的内容进行修改。

从软件工程客观角度看，软件的安全性隐患又源于以下几个方面：

1) 软件复杂性和工程进度的平衡。软件复杂性提高，不仅是编码工作量的提高，更重要的是其中需要考虑的问题更复杂，测试用例规模也呈指数级增长。但工程进度只是按照软件规模进行适当延长，很多问题没能解决便投入使用。

2) 安全问题的不可预见性。主要是软件工程师不了解运行的实际情况，在测试时做出过于简单的假设。无法安全考虑一部分问题，包括对软件的功能、输出和软件运行环境的行为状态，或者外部实体(用户、软件进程)的预期输入。攻击者有足够的时间进行攻击方法的研究。

3) 由于软件需求的变动。软件规格说明书或设计文档无法一开始就确定下来；在现代软件工程中，很多软件的需求变动，导致其设计本来就是变动的，很多安全问题可能在变动的过程中被忽略。

4) 软件组件之间交互的不可预见性。如客户在使用软件的过程中，自行安装第三方提供的组件，开发者根本无法知道客户的软件将和谁交互，软件在运行过程中可能会出现安全问题。

可见软件的安全问题无法完全避免。即使在需求分析和设计时可以避免，或者在开发时可以避免，但缺陷还是会在软件汇编、集成、部署和运行时被引入。不管如何忠实地遵守一个基于安全的开发过程，只要软件的规模和复杂性继续增长，一些可被挖掘出来的错误和其他缺陷是必然存在的。因此，安全编程技术不是为了消除安全隐患，而是为了尽量减少安全隐患。

### 1.2.3　软件缺陷和漏洞

#### 1. 缺陷和漏洞的定义

软件存在的各种问题，中文通常用“软件缺陷”一词，但在英文中有多个词与之对应。例如缺陷(bug)、缺点(defect)、偏差(variance)、谬误(fault)、失败(failure)、矛盾(inconsistency)、错误(error)、毛病(incident)、异常(anomy)等。IEEE 标准 729-1983 中对软件缺陷给出了一个标准定义：从产品内部看，软件缺陷是软件产品开发或维护过程中存在的错误、缺点等各种问题。从外部看，软件缺陷是系统所要实现的某种功能的失效或违背。互联网词汇将漏洞定义为：系统设计、实现和操作管理中存在的缺陷和弱点，它们可以被用来违背系统的安全策略。

#### 2. 软件缺陷存在的原因

软件缺陷产生的主要原因有：需求规格说明书(Requirement Specification)包含错误的需求，或漏掉一些需求，或没有准确表达客户需要的内容，需求规格说明书中有些功能不可能或无法实现；系统设计中的不合理性；程序设计中的错误；程序代码中的问题，包括错误的算法、复杂的逻辑等。

软件缺陷的主要类型有：功能、特性没有实现或实现不完全；设计不合理，存在缺陷；实际结果和预期结果不一致；运行出错，包括运行中断、系统崩溃；数据结果不正确、精度不够。

软件缺陷的严重性级别可分为 4 类：

- 致命(fatal)缺陷：造成系统和应用程序崩溃、死机、系统挂起，数据丢失、主要功能完全丧失等；
- 严重(critical)缺陷：功能或特性没有实现，主要功能丧失，导致严重问题或致命的错误声明；
- 一般(major)缺陷：不影响系统的级别使用，但没有很好地实现功能，没有达到预期效果，如次要功能丧失、操作时间长等；
- 轻微(minor)缺陷：对功能几乎没有影响，产品及属性仍可使用。

软件缺陷除了严重性外，还存在不同的状态来反映软件缺陷处于一种什么样的状态，以便跟踪和管理某个软件产品的缺陷。激活(active)状态：问题还没有解决，新发现的缺陷或验证后仍存在的缺陷。已经修正(fixed)状态：针对所存在的缺陷，修改程序，认为已经解决问题或通过单元测试。非激活(inactive)状态：测试后验证缺陷不存在的状态。

#### 3. 存在软件安全漏洞的原因

根据漏洞出现的阶段，软件安全漏洞可分为两大类：设计漏洞和实现漏洞。设计漏洞是软件设计阶段发生的设计错误，无论编码人员如何实现，都存在安全问题。实现漏洞是由软件编码中的安全缺陷造成的，如没有检测返回代码、没有正确定义缓冲区的大小、没有正确处理非预期的输入等。

常见的设计漏洞有如下两个方面：一是密码技术使用不当，如选择错误的密码学方法(对称密码算法、非对称密码算法、哈希函数等)来完成加密、完整性检验和认证等。在使用密码技术时，没有使用标准化的成熟算法，如 AES、RSA 等，而是自创一套未得到公开研究和验证的密码算法。即使是使用现有的成熟算法，也没有使用较知名的密码库，如 Microsoft Crypto

API 或 OpenSSL，而是自己编写的或来源于不知名的密码库。二是安全需求不明确，在系统分析阶段没有明确安全的需求，对威胁没有明确的建立模型。

### 4. 软件漏洞产生机理

软件漏洞的产生机理有以下 3 个方面：

#### 1) 应用程序的内存布局

执行应用程序时，可执行的应用程序以及提供支持的库被载入内存。即使系统的物理内存很小，每个应用程序也被分配 4GB 的虚拟内存，即基于 32 位的地址空间。当应用程序运行内存管理时，内存管理自动将虚拟地址映射到真实数据所在的物理地址。

内存可分为用户模式和内核模式。用户模式内存一般是载入和执行应用程序的内存区域，而内核模式内存是载入和执行内核模式元件的内存区域。因此，应用程序不能直接访问任何内核模式内存，否则会造成非法访问。如果应用程序需要对内核模式内存进行正当访问，就在操作系统和应用程序间进行用户模式和内核模式的转换。

默认情况下，2GB 的虚拟内存空间被提供给用户模式，另外，2GB 的虚拟内存空间被提供给内核模式。这样，0x00000000 至 0x7FFFFFFF 这一段提供给用户模式，0x80000000 至 0xBFFFFFFF 这一段提供给内核模式。

一个可执行程序不仅和它需要的应用程序的动态链接库链接，也和默认的系统堆一起共享用户模式的地址空间。在不同主机上，只要操作系统的版本和应用程序是一样的，都将应用程序的动态链接库载入相同的内存地址。

#### 2) 应用程序结构

所有应用程序被载入 3 个主要的内存区：堆栈段(Stack Segment)、数据段(Data Segment)以及代码和文本段(Code and Text Segment)。堆栈段存储了局部变量和程序调用，数据段存储了静态变量和动态变量，文本段存储了程序指令，数据段和堆栈段对于每个程序都是私有的，即其他应用程序不可以访问这些区域。另一方面，文本段部分是只读段，这是其他程序可以访问的。然而，如果试图向这个区域进行写入操作，会引起非法段操作(Segment Violation)。

#### 3) 进程的内存布局

每个进程的虚拟内存被分为内核地址空间和用户地址空间。Windows 以及 Linux 中的用户地址空间包括堆栈段、堆地址空间、程序代码和各种其他段，如 BSS 编译器放在静态数据的段。在 Linux 系统上，堆栈处于内存地址的高端，堆栈的顶端仅比 0xc0000000 低 1 比特。在 Windows 系统上，这一点完全不同，堆栈位于内存地址的低端，所以通常地址的最重要字节(MSB)通常为 0。且内存的设置更复杂，进程可以有许多堆，每个动态链接库都有各自的堆和堆栈。这个区别通常使 Windows 上堆栈溢出攻击的利用比 Linux 上的利用更困难。因为简单的基于堆栈的 shellcode 在它的主体中至少含有一个来自堆栈的地址，但是字符串 copy 函数将在 0 字节时停止复制，无法复制全部的 shellcode，这就是著名的 NULL 字节问题。

### 1.2.4　针对软件安全问题的测试

#### 1. 软件测试概述

IEEE 对软件测试给出的定义是："使用人工或自动手段来运行或测定某个系统，其目的在于检测该系统是否满足规定的需求，或者弄清预期的结果与实际结果的差别。"因此，软件测试的目的是发现软件中的错误，并在交付用户使用前消除这些错误，这几乎成为一个公认的概念。这里的"错误"实际上是一个广义的概念，初学者往往将其理解为"编码错误"，实际上，能引起软件错误的因素很多，绝不仅是编码方面的原因，包括很广泛的内容：软件的需求分析者曲解了用户的需求，测试时发现实现的流程和用户的叙述不一样；软件的设计者在设计时没有考虑某些现场因素，导致软件在真实环境下测试时无法正常运行；软件编码者的疏忽，将某些逻辑流程写错，使得程序得不到预想的结果等。

由此可见，软件测试的根本目标是尽可能多地发现并排除软件中潜藏的错误，最终把一个高质量的软件系统交给用户使用。测试的目的包括以下几部分：测试是为了发现程序中的错误而执行程序的过程；优秀的测试方案是尽可能发现目前尚未发现的错误的测试方案；成功的测试方案是找到目前尚未发现的错误的测试。

不过，如果将软件测试效果的评价指标定义为查出错误的个数，认为无法检测错误的测试就是没有价值的测试，则是片面的观点。因为没有发现错误，或者发现错误较少的测试，侧面反映出软件质量较高。因此，测试同时也是评定软件质量的一种标准；发现很多错误的测试，不一定是成功的，如果软件本身质量较低，那么不能通过发现错误的个数越多，得出软件剩下的错误越少的结论；已经发现的错误很多，可能未知的错误也很多。

从另一角度看，通过软件测试找到错误，除了能够解决错误外，还可以通过分析错误产生的原因和错误的发生趋势，帮助软件的开发者发现当前软件开发过程中的缺陷，以便及时改进；另外，通过对错误进行分析，也可以帮助测试人员设计出更加具有针对性的测试方法，提高测试工作的效率和效果。

软件测试的意义主要体现在减少软件中的错误。通过软件测试可以发现软件中存在的错误，通过完全地修改这些错误，可以减少软件中的错误，提高软件的可靠性。通过软件测试，对发现的错误进行分析和统计，可以评估软件的综合性能。当然，即使软件测试没有发现任何错误，也可以作为评估软件综合性能的手段等。

从实际项目的测试工作划分，软件测试工作可分为以下几个过程：

- 单元测试：对软件的每一个程序单元进行测试，检查各个程序模块的正确性；并配合适当的代码审查；
- 集成测试：把已测试过的模块组装起来，以便发现与接口有关的问题，如数据模块间传递、模块组合性能、模块调用性能等；
- 确认测试：检查软件是否满足需求规格说明书中的各种需求，以及软件配置是否完全、正确；该测试又叫做验收测试，目的是验证软件的有效性；
- 系统测试：把已经通过验收的软件，放入实际运行环境中运行；用户记录在测试过程中遇到的一切问题，定期报告给开发者。

这几个测试过程，从软件开发生命周期的一开始就应该执行，而且贯穿软件工程中的每个阶段。

软件测试的方法有很多种分类，第一种是分为静态测试方法和动态测试方法。

静态测试方法指不实际运行被测试的软件，对软件进行分析、检查和审阅来寻找逻辑错误。主要工作包括：对需求规格说明书、软件设计说明书、源程序进行检查和审阅；检查以上工作是否符合标准和规范；通过结构分析、流图分析等方法，指出软件缺陷；对各种文档进行测试等。静态测试是软件开发中十分有效的质量控制方法之一。该方法在软件开发生命周期的早期和中期阶段尤其有效。由于程序还没有编出来，可以直接运行的代码尚未产生，又必须对设计的一些思路进行检查或审核，因为初期的工作质量可能直接关系到软件开发的成本。因此，在这些阶段，可大量采用静态测试方法。静态测试主要靠人工来完成，不过近些年来，也开发了不少自动化工具，进行计算机辅助测试。但短期内想要实现测试的自动化难度较大。静态测试的质量更多取决于测试的组织和测试者的水平，定性地分析软件质量的情况居多，有一定的局限性。

动态测试和静态测试不同，它指的是在测试的过程中，实际运行软件，检测软件的动态行为和运行结果的正确性。动态测试包括两个基本要素：一是被测软件；二是在软件运行过程中的输入数据，每一次测试需要的测试数据叫做测试用例。因此，动态测试一般在软件编码阶段完成之后进行。动态测试由于其较强的错误检测能力，受到广泛采用。动态测试的过程是：设计测试用例，输入到程序中；看预期结果和实际运行结果是否一样；得出最终结论。动态测试方法中，最大的难点是测试用例的设计，因为穷举性测试无法实现。

另一种分类方法是从对程序内部结构的可见性来分，分为黑盒测试和白盒测试。

黑盒测试又称为功能测试。用该方法进行测试时，把被测程序当作黑盒，测试者无须知道程序内部结构，只需要知道程序的输入以及输出是否和预期输出相符。用例设计方法包括等价类划分法、边界值分析法和因果图法等。

白盒测试又称结构测试或逻辑驱动测试。用该方法进行测试时，测试者必须了解被测程序的内部结构，根据被测程序的内部构造设计测试用例。在白盒测试的过程中，需要测试用例的设计对被测程序的结构做到一定程度的覆盖。常见的测试用例设计方法有基本路径法、条件测试法和循环测试法等。

将软件测试划分为静态测试和动态测试与划分为黑盒测试和白盒测试是没有矛盾的，两种方法可以互相渗透与融合。一般情况下，静态测试只利用白盒测试法，动态测试则使用了黑盒测试与白盒测试；从另一个角度看，黑盒测试一般都用于动态测试，而白盒测试一般可以用于静态测试和动态测试。

### 2. 软件安全测试的必要性

安全测试是在充分考虑软件安全性问题的前提下进行的测试，普通的软件测试的主要目的是：确保软件不会去完成没有预先设计的功能，确保软件能够完成预先设计的功能。但是，安全测试更有针对性，同时可能采用一些和普通测试不一样的测试手段，如攻击和反攻击技术。因此，实际上，安全测试就是一轮多角度、全方位的攻击和反攻击，目的就是要抢在攻击者之前尽可能多地找到软件中的漏洞，以减少软件遭到攻击的可能性。

安全测试基于软件需求说明书中关于安全性的功能需求说明，测试的内容主要是：软件的安全功能实现是否与安全需求一致。通常情况下，软件的安全需求包括：数据保密和完整可用；通信过程中的身份验证、授权、访问控制；通信方的不可抵赖；隐私保护、安全管理；

软件运行过程中的安全漏洞等。

以 Web 站点为例，需要考虑的方面包括程序本身的安全、系统安全、数据库安全和性能安全。其中程序本身的安全内容包括用户权限划分是否得当、用户权限改变是否会造成混乱、用户数据是否会混淆、用户密码是否可以用某些手段得知、系统可否有后门登录、是否进行了 session 检查、是否有 SQL 注入和跨站脚本等隐患。系统安全的内容包括服务器是否存在漏洞被实施 DDoS 攻击、是否可能被攻击者注入木马、操作系统是否安全、防火墙和杀毒软件是否齐全有效等。数据库安全的内容包括系统数据是否机密、系统数据是否完整、系统数据是否进行了很好的权限控制、系统数据可备份和恢复能力如何等。性能安全的内容包括系统是否能够保证每周 7 天、每天 24 小时连续工作，用户访问应用服务器是否进行了优化以及对数据库的访问是否实现了优化。

因此，软件安全测试和一般测试有很大的区别。一般测试主要是确定软件的功能能否达到，如果没有达到，就进行修改，其任务具有一定的确定性。但是，安全测试主要是检查软件实现的功能是否安全可靠，需要证明的是软件不会出现安全方面的问题，如数据篡改、非授权访问、遭受 DDoS 攻击等。

### 3. 软件安全测试的过程

软件的安全测试，一般根据设计阶段的威胁模型来实施。应该从设计阶段就开始考虑安全问题，设计要尽可能完善，可采用威胁建模的方法在软件设计阶段加入安全因素的考量。威胁建模过程一般如下：在项目组中成立一个小组，该小组中的人员是项目组中对安全问题比较了解的人；站在安全角度，分解系统的安全需求；确定系统可能面临的威胁，将威胁进行分类，可以画出威胁树；选择应付威胁或缓和威胁的方法；确定最终解决这些威胁的技术。既然在设计阶段，就将系统可能出现的一些安全问题写在文档里了，因此安全测试也应该基于这些内容。软件安全测试的过程可分为以下几个步骤：

1) 基于前面设计阶段制定的威胁模型，设计测试计划。该过程一般基于威胁树，以用户口令安全问题威胁树为例，测试计划可基于口令安全可能遭受的各个攻击进行制定。

2) 将安全测试的最小组件单位进行划分，并确定组件的输入格式。实际上，与传统的测试不同，威胁模型中，并不是所有的模块都会有安全问题，因此只需将需要安全测试的某一部分程序取出来进行测试，对安全测试的最小组件单位进行划分。此外，每个组件都提供了接口，也就是输入，在测试阶段，测试用例需要进行输入，这就必须将每个接口的输入类型、输入格式等都列出来，以便制定测试用例。这些输入的例子有 Socket 数据、无线数据、命令行、语音设备、串口和 HTTP 提交等。

3) 根据各个接口可能遇到的威胁或系统的潜在漏洞，对接口进行分级。在该步骤中，主要是确定系统将要受到的威胁的严重性，将比较严重的威胁进行优先测试，对严重性的判断应该源于威胁模型。可以通过很多方法对接口受到的威胁性进行分级，例如存在一种积分制方法，对各个接口可能受到的各种威胁进行积分，最后累加，优先测试那些分数排在前面的接口。

4) 确定输入数据，设计测试用例。每个接口可以输入的数据都不相同，由于安全测试不同于普通的测试，因此还要更加精心地设计测试用例。有时候还要精心设计输入的数据结构，如随机数、集合等的设计，都必须是为安全测试服务的。在测试用例的设计过程中，安全测

试实际上是对程序进行的安全攻击。因此，不但数据本身需要精心设计，测试方案也要精心设计。如在对缓冲区溢出的测试中，必须精心设计各种输入，从不同方面对程序进行攻击。如上面网站中可以设计的测试用例内容如下：针对用户权限划分的测试，使用各种权限登录并操作的方案；针对用户密码的测试，使用猜测密码，查看数据库密码保存的方案；针对系统可否有后门登录的测试，使用尝试各种登录方法的方案；针对 session 检查的测试，使用不登录进行操作的方案；针对 DDoS 攻击的测试，使用反复进行 DDoS 攻击的方案；针对木马注入的测试，使用注入木马的方案；针对防火墙和杀毒软件的测试，使用注入病毒的方案；针对系统数据可备份和恢复能力的测试，使用数据破坏的方案；针对保证每周 7 天、每天 24 小时连续工作的测试，使用持续运行足够时间的方案。

5) 攻击应用程序，查看效果。用设计的测试用例攻击应用程序，使得系统处于一种受到威胁的状态，得到输出。

6) 总结测试结果，提出解决方案。本过程中，将预期输出和实际输出进行比较，得出结论，写出测试报告，最后提交相应的人员，解决错误。

以上是一般的测试过程。近年来，在软件安全性测试领域还出现了一些研究成果，使用计算机来进行自动测试，这些成果主要包括以下 4 个方面：

- 用形式化方法执行安全测试，该方法用状态迁移系统描述软件的行为，将软件的功能用计算逻辑和逻辑演算来表达，通过逻辑上的推理和搜索来发现软件漏洞。
- 基于模型的安全功能测试，在该方法中，首先对软件的结构和功能进行建模，生成测试模型，然后利用测试模型导出测试用例。该方法成功与否取决于建模的准确性，对身份验证、访问控制等情况下的安全测试比较适用。常用模型有 UML 模型和马尔可夫链模型等。
- 基于输入语法进行测试。接口的输入语法，定义了软件接受的输入数据的类型、格式等。该类方法中，首先提取被测接口的输入语法，如命令行、文件、环境变量、套接字等，然后根据这些语法，生成测试用例。此类测试方法比较适用于被测软件有较明确的接口语法的情况，范围较窄。
- 采用随机方法进行测试。该方法又称为模糊测试，将随机的不合法数据输入到程序中，有时能发现一些意想不到的错误，在安全性测试中越来越受到重视。

软件测试是软件工程中研究比较活跃的一个分支，针对安全测试的研究也受到较多学者的重视。有关一些安全测试方面的最新进展，读者可以参考相关文献。比较热门的方向包括权限系统的自动测试、形式化方法对测试用例的表达、分布式环境下的测试和云计算环境下的测试等。

### 4. 安全审查

安全审查是指对软件产品进行安全方面的人工检查，是软件质量保证的一个重要环节，主要包括代码的安全审查、配置复查和文档的安全审查。

#### 1) 代码的安全审查

代码的安全审查是审查小组人工测试源程序的过程，是一种非常有效的程序安全验证技术，在代码的安全审查过程中，首先组建代码的安全审查小组，最好由如下人员组成：组长，应该是一名很有能力的程序员；程序的设计成员；程序的编写成员；程序的测试成员。

代码安全审查的步骤如下：小组成员先研究设计说明书，力求理解软件的设计，然后重点针对威胁模型进行讨论；由设计者介绍威胁模型中的一些细节；程序的编写者逐个模块地解释是怎样用程序代码解决威胁模型中提出的解决方案的；对照安全程序设计常见错误，分析审查程序；发现错误时记录错误，继续审查。

**2) 配置复查**

软件配置实际上是软件需求规格说明、软件设计规格说明、源代码等的总称，配置复查实际上是软件验收测试的重要内容。配置复查需要保证如下内容：软件配置的所有成分都齐全；软件配置质量符合要求；文档与程序完全一致；具有完成软件维护所必须准备的细节；使用手册的完整性和正确性等。

软件配置复查的过程中，必须仔细记录发现软件安全测试过程中的安全遗漏或错误，并且适当地补充和改正。

**3) 文档的安全审查**

文档在软件工程中非常重要。对用户来说，软件事实上就是文档，因此文档是影响软件质量的决定因素，有时可以说，文档比程序代码更重要。在文档中，关于安全问题的描述不能忽视，必须进行审查。软件系统的文档可分为两类：用户文档和系统文档。

用户文档主要描述系统功能和使用方法，而并不关心这些功能是怎样实现的。用户文档是用户了解系统的第一步，它应该能使用户获得对系统的准确的初步印象。用户文档至少应该包括下述 5 方面的内容：

1) 功能描述，说明系统的功能；

2) 安装文档，说明怎样安装这个系统，怎样对系统进行配置，使其适应特定的运行环境；

3) 使用手册，说明系统使用方法，用户应该可以通过这个文档学会系统的使用，可通过例子或图形化方法表达这些问题；

4) 参考手册，详尽描述软件中提供给用户使用的所有系统设施及其使用方法。另外，参考手册中还应该解释系统可能产生的各种输出信息，如出错信息的含义；

5) 操作员指南，指示操作员应该如何处理使用过程中出现的一些情况。用户文档可分别设立独立的文档，也可以设置为大文档的各个分册，具体做法根据系统的复杂性确定。

系统文档指从问题可行性分析、问题定义、需求分析、系统总体设计、详细设计到测试和测试报告这样一系列与工作有关的文档。系统文档描述了系统从设计到实现，最后到测试的过程。该部分的文档包括可行性研究报告、需求分析说明书、总体设计说明书、详细设计说明书、测试计划和测试报告。文档的安全审查是针对软件项目中的安全问题进行的审查，此过程中，主要进行的工作是、根据威胁模型，在各个文档中审查是否进行了良好的表达。比如：用户文档中是否含有解决安全问题相关的措施，来提示用户的操作尽可能保证安全性；系统文档中是否对所有安全问题进行了解决，并用清晰的表达方式描述出来。

## 1.3 软件安全相关标准

在项目起始阶段，必须考虑软件需要遵守的安全规则和规章。在项目实施阶段，必须遵

循软件安全的原则。本节最后将介绍与软件安全相关的国际标准。

### 1.3.1　安全规则与规章

一些相关规章包括：符合 Web 应用程序安全标准(WASS)，服从 HIPPAA 隐私和安全规章，或满足联邦信息安全管理法案(FISMA)标准，或需要满足支付卡行业数据安全标准(Payment Card Industry Data Security Standard)，或遵守金融现代化条例(The Graham-Leech-Bliley)以及 GRC(Governance Risk management and Compliance)、PCI(Payment Card Industry)、SOX(Sarbanes-Oxley Act，SOX 法案)相关标准。

#### 1. OWASP 的 WASS

OWASP(开放式 Web 应用程序安全计划组织)制定了一些安全标准，如 ISO 17799，即信息安全管理国际标准，这是一个由国际标准化组织发布的、便于采用和理解的标准。这项标准却极少具体应用于与管理安全的 Web 站点相关的工作中。在实现了一个安全的 Web 应用程序后，信息安全管理就不可避免。虽然 ISO 17799 在标识应考虑的策略和过程方面做出了卓越的贡献，但这一标准并没有解释这些策略和过程应该如何实现，也没有给出相关的工具。它只是关于应该考虑哪些策略和过程的指南，并且没有强制要求实现提到的所有策略和过程。

OWASP 还推荐了以下安全标准：Web应用程序安全标准(WASS)，计划组织创建一组最小化集合的提案，要求那些处理信用卡信息的 Web 应用程序必须体现出这些要求。本计划的目标是开发“专用的”、“可测试的”准则，这些准则要能作为独立存在的安全标准，或者集成到现有的安全标准，比如集成到持卡人信息安全规划(Cardholder Information Security Program，CISP)，这是一个厂商中立及技术中立的标准。通过对照这一标准进行测试，能够判定在某个基于 Web 的应用程序开发中，是否遵循了最低限度的安全过程，是否坚持遵守了最佳实践。

#### 2. HIPPA

HIPPA/1996 法案是美国国会于 1996 年制定的。根据医疗保险制度和医疗服务中心的网站信息，HIPPA 的目的之一是保障工人在更换工作或失去工作时他们及其家人的健康保险。HIPPA 的目的之二是为了简化管理，即为电子健康事务和国家范围内的身份识别建立国家标准，以服务于供应商、健康保险计划和雇主，帮助人们保护健康信息的私密性，在简化管理的同时还解决健康数据的安全性和私密性。这些标准是为了改善国家卫生保健体系的效果和可行性而鼓励在美国的保健体系中广泛使用电子数据交换。

美国公民权利办公室负责执行的 HIPPA 隐私规则保护个人标识健康信息的私密性；HIPPA 安全规则是为电子保护的健康信息安全而设立的国家标准。HIPAA 隐私规则规定，联邦政府个人健康信息由相关机构来保护，并提供一系列关于这些信息的权利。同时，隐私规则兼顾了当个人健康信息对病人的医疗和其他重要用途有帮助时可以被披露。安全规则规定了一系列为相关机构提供关于行政保护、物理性保护和技术性保护的规则，以确保电子保护信息的机密性、完整性和可用性。

管理(administrative)规则定义了实体应该遵守的策略和过程，例如相关机构(必须遵守 HIPAA 要求的机构)必须保护隐私并指派一名可以对开发负责和能实现所有需求策略的隐私

官员。程序应该明确员工或员工的等级来确定谁可以访问受保护的电子健康信息(EPHI, Electronic Protected Health Information)。访问 EPHI 必须只限于那些需要用其来完成工作职能的雇员。这些程序必须处理访问授权、编制、修改和终止。物理(physical)规则定义了如何控制物理访问来阻止对保护数据的不恰当的访问，例如控制机构必须支配从网络中引入和移除硬件和软件(当设备老化时必须妥善处理，以确保 PHI 不泄密)。当有权使用包含健康信息的设备时应该审慎控制和监督。访问硬件和软件时，必须确保是正确的授权者。

技术(technical)规则定义了如何控制对计算机系统的访问，以及实体如何保护在公共网络上传输的包含 PHI 信息的通信，以阻止通信被非预想的接收者之外的攻击者劫持。例如 PHI 信息系统所在地点必须进行妥善保护，以免入侵。当信息在网络上公开传送时必须利用某种加密方式进行加密。如果网络系统是封闭的，必须充分考虑以及提供可选的加密方式。任何相关机构必须负责确保自己系统的数据没有被以未授权者的形式改变或擦除。数据佐证，包括利用校验和、double-keying、消息认证和数字签名可以用来确保数据的完整性。相关机构必须验证其通信的实体。认证包括证实，一个实体是谁声明的。佐证的例子包括密码系统、双向或三方握手系统、电话回拨和令牌系统。

### 3. FISMA

联邦信息安全管理法案(FISMA, Federal Information Security Management Act of 2002)是美国联邦法 2002 年以第三版 E-Government 的标题颁布，该法案承认信息安全对国家经济和国家安全利益的重要性。该法案要求每个联邦机构为信息和信息系统的开发、文件和实现机构提供信息安全保护，保障其运营和资产，其中包括提供者或另一个机构、承包人或其他资产。FISMA 已经引起了联邦政府对网络安全的注意，并明确强调了以风险为基础的成本效益安全政策。FISMA 要求代理官员、信息安全主管和监察长进行年度安全程序审查并把结果向管理和预算办公室做报告(OMB，Office of Management and Budget)。OMB 使用这些数据来协助其监督职责并用其编写年度报告给国会，2008 年联邦政府在安全方面总共花费 62 亿美元。

#### 1) 法案的目的

FISMA 负责把各种责任分配给各联邦部门。美国国家标准技术研究院(NIST)以及管理和预算办公室(OMB)负责加强信息系统安全。特别是，FISMA 要求各个机构领导执行政策和程序时以一种低开销、及时和有效的方式把风险控制在可接受的范围之内。按照 FISMA，信息安全措施意味着保护信息和信息系统免受未授权者的侵入、利用、披露、破坏、修改以保障其完整性、保密性和可用性。

#### 2) FISMA 的实现

与 FISMA 一致，NIST 负责开发标准、指导方针以及相关联的方法和技术，为所有部门运营和资产提供足够的信息安全保障，不包括国家安全系统。NIST 与联邦机构密切配合来提升他们对实现 FISMA 的理解，从而保护它们的信息和信息系统，并且提供强大信息安全基础的标准和指导方针。NIST 通过计算机安全部门的信息技术来履行其法定职责。

#### 3) FISMA 项目实施

国家漏洞数据库(NVD)是美国政府为自动化信息安全计划(ISAP)准备的内容仓库，数据库中的数据能帮助实现漏洞管理、安全测量和服从的自动化。

## 1.3.2 软件安全的原则

类似密码学领域的安全原则 Kohoff: 密码系统的安全性应该建立在只是对密钥的保密之上，即假定攻击者是知道加密算法的。软件安全方面也有一些相应的指导原则。

**原则 1：让最弱的环节变得安全**

首先使得安全性最薄弱的环节变得安全。安全实践者通常认为：安全是一个链条，其安全性由最弱之处决定。软件的安全由最弱的组件决定。攻击者选择系统中最弱的地方攻击，因为那里最容易攻破。攻击者最可能攻击一个软件系统中的一个弱点，而不是试图穿透一条坚固的防线。例如：一些密码算法需要很多年的时间攻破，因此攻击者不会攻击一个网络中的加密信道，取而代之，一个通信端点可能更容易攻破。知道软件弱点何时被加固，意味软件开发者知道软件能否足够安全地发布。

**原则 2：实践纵深防御**

分层的安全防御措施能够减少攻击的成功概率。使用冗余的安全机制使得攻击者难以攻克所有的防御机制。例如：软件系统使用认证检查来防止攻击者穿透防火墙。使用多层防御能够阻止单一失效点的发生，单一失效点往往也是承担所有安全防御任务的安全焦点。

**原则 3：安全的错误退出**

当系统发生错误时，系统应该安全地退出。通常包括以下几点：安全的默认设置(如停止被访问)；撤消前面的操作直到恢复到安全状态；检查错误返回值等。系统的私密性和完整性应该继续保持，即使是可用性已经被破坏。在系统失效期间，攻击者必须不被允许访问需要特殊权限访问的对象。由于系统在失效期间给潜在攻击者透露的敏感信息可能导致新的攻击，因此必须正确定义系统失效时的行为。任何复杂的系统都应该有错误模式。错误是无法避免的，应该事先计划的是当错误发生时哪些安全问题需要避免。当有很多系统发生各种错误时，可能表现出不安全的行为。这样，攻击者只需要导致错误，或者等待错误的发生，然后就可以根据系统在错误处理时的不安全行为发起攻击。

**原则 4：最小权限原则**

仅将所需权限的最小集授权给需要访问资源的主体，并且权限的持续时间应该尽量短。给用户不必要的(过多的)权限通常导致信息被泄漏和不必要的改变。因此，精准的访问权限分配将减少对系统的破坏。

**原则 5：权限分离原则**

权限分离原则的解释是：可行的情况下，保护机制需要两个钥匙来开一把锁将比使用一把钥匙要更加可靠和灵活。这一观察早在 1973 年便由 R. Needham 提出。基本思想是：一旦被锁，这两把“钥匙”可以是由物理上分离的或是完全不同的程序来负责。从此将没有单一的偶然事件、伪装或受托人单方违背而造成对保护信息的损害。这条原则经常用于银行安全保险箱，同时在核武器防御系统上也有应用，只有两个不同的人同时给出了正确的命令时导弹才能被发射。在计算机系统中，分离“钥匙”是指当两个或两个以上的条件满足时访问才能被允许。系统在授权访问某个对象时应该确保多个条件得到满足，通过检查单一条件来决定是否访问对强安全性来说不够充分。如果攻击者能够获得一个特权但没有第二个，将不能

发起有效攻击。如果软件系统主要由一个组件组成，那么使用多种检查措施访问不同组件的想法将无法实现。

**原则 6：保护机制的经济性**

评价系统安全时的一个因素是复杂性。如果设计、实现或安全机制非常复杂，那么安全弱点存在的可能性将会增加。复杂系统中的微妙问题可能很难发现，特别是在有大量代码存在的情况下。一个策略是使用控制点来简化代码，使用公共的功能函数能减少代码数量，程序设计者和开发人员都应该设法使系统尽量简化。尽可能考虑重用组件，只要被重用的组件是高质量的就建议这样做。如果成功地应用组件，代码量就会减少。这种思想特别适用于密码库。在已有大量可用的密码库的情况下，不必重新实现 AES 或 SHA-1。被广泛使用的密码库要比组织内部使用的密码库更健壮。当然，即使是被广泛使用的组件，也可能出现问题。

**原则 7：最小共性机制**

避免多个主体通过共享机制有权使用同一资源。例如，互联网上的服务应用程序同时允许攻击者和用户获得访问权。敏感信息可能通过这种机制在不同用户之间共享。不同的机制对不同的主体提供灵活的访问控制，防止只有一种机制实现时的可能出现的安全问题。

**原则 8：勉强信任原则**

开发者应该认为他们系统所在的环境是不安全的。无论是外部系统、代码、人员等，都应该被时刻关注。当开发一个应用程序时，软件工程师应该预先考虑未知用户输入的所有可能情况。即使用户是已知的，也要怀疑其可能发起社会工程攻击，也要把他们的威胁加以考虑，没有系统是完全安全的，所以两个系统之间的接口应该紧固。最小化对其他系统的信任可以加强自身应用程序的安全性。

**原则 9：秘密的不安全性假设**

应该始终假设，攻击者能够获得关于系统的足够信息并且发起攻击。汇编和反编译器等可以帮助攻击者获得可能存储在二进制文件中的敏感信息。同样，内部攻击可能是偶然的或恶意的，并将导致敏感信息被利用。使用成熟的保护机制来保证敏感信息的机密性。

**原则 10：完全仲裁**

当主体请求访问时，软件系统需要检查每个对象，特别是关系重大的对象，减小错误地授权导致主体权限提升的机会。系统检查主体对象的访问许可仅有一次的话，将被攻击者利用，应该在每次访问时都检查。如果主体的访问控制权限在第一次授权后减少了，并且系统没有在第二次访问时检查，就发生了一次访问权限违例。短时间保存权限信息可以提高系统性能，但可能付出越权访问的代价。

**原则 11：心理接受能力**

安全机制不应该阻止资源的可访问性，如果安全机制阻碍资源的可用性和可访问性，用户可能选择关闭这些机制。在可能的情况下，安全机制对系统用户应该是透明的或只有极小的使用阻碍。安全机制应该对用户很“友好”，从而让用户便利使用，方便用户对程序的理解。

**原则 12：保护隐私**

保护软件私密信息免受攻击者窃取是软件安全一个非常重要的方面。如果攻击者破解一个软件系统并盗取客户的私密信息，用户会对这个软件系统失去信心。攻击者也可以攻击敏感系统信息，从而提供给其他攻击者。防止攻击者访问私密信息或混淆这些信息可以降低信息泄漏的危险。

### 1.3.3　软件安全相关的国际标准

软件质量标准：ISO 9000 系列标准是 ISO 国际标准化组织 176 技术委员会(TC/176)制定的国际标准，其核心思想是质量保证标准 ISO 9001/2/3 和质量管理标准 ISO 90040，前者是“需方对供方要求质量保证”的标准，后者是用于“供方建立质量保证体系”的标准。ISO 9000 标准是为制造硬件产品而制定的标准，ISO 试图修改 ISO 9001 用于软件开发方面，但效果不佳。于是，另行制定出 9000-3 标准，用于软件开发、供应及维护的指南。我国制定了 CB/T 19000 国家标准，该标准等同于 ISO 9000 标准。这两个标准在内容上基本上是对应的。

ISO 9000-3 标准是 ISO 质量管理和质量保证标准在软件开发、供应和维护中的使用指南，并不作为质量体系注册、认证时的评估准则，主要考虑软件行业的特殊性制定的标准，其核心内容包括：合同评审、需方需求规格说明、开发计划、质量计划、设计和实现、测试和确认、验收、复制、交付和安装维护。

ISO/IEC 9000-3 是国际标准化组织国际电工技术委员会用于软件产品和相关服务的质量管理标准(Quality Management Standard)，新的 ISO IEC 9000-3 2004 标准于 2004 年 2 月 15 日公布，原有的 ISO 9000-3 1997 软件标准作废。

ISO/IEC 9000-32004 是软件工程标准 ISO 9001 2000 的指导标准。它提供对应用 ISO 9001 2000 标准的指导，例如获得、供应、开发、操作和维护计算机软件相关支持服务。它并没有增加或改变 ISO 9001 2000 的要求，应用该标准对包括商业软件、组织软件、硬件嵌入软件以及相关软件服务等是合适的。有些组织可能参与所有这些行为，有的组织只是参与某个领域。无论哪种情况，组织的质量管理系统都应该搜索所有的软件相关以及不相关的方面。该标准确定了应该解决的一些问题，这些问题独立于该组织采用的技术、生命周期模型、开发过程、行为次序以及组织结构。软件工程标准 ISO/IEC JTC 1/SC7 通常作为补充和参考，这些标准包括 ISO/IFX 12207、ISO/IEC TR 9126、ISO/IEC 14598、ISO/IEC 15939 和 ISO/IEC TR 15504。另外，早在 1996 年，美国国防部颁发了安全编程标准：MIL-STD482B 系统安全编程需求(System Safety Program Requirements)。

## 1.4　本章小结

本章简要介绍了软件安全问题的现状，包括恶意软件、漏洞、威胁等的统计数据，明确了软件安全的概念和研究的内容，并从多个方面(如软件工程、软件保证、软件质量、软件可靠性等)介绍了与软件安全相关的领域，以帮助读者明确软件安全的内涵和外延。本章还介绍了一些基本概念和专有名称，用于相关背景的复习，并介绍了常用的安全工具和针对软件安全问题的测试方法。最后介绍了软件安全相关标准，包括安全规则与规章、软件安全的原则和软件安全相关的国际标准。

# 第2章　软件安全设计与编程

## 本章导读

本章主要介绍计算机软件安全的设计流程、安全的软件开发周期，并阐述了规范的安全软件在不同的开发阶段需要的不同侧重点，详细描述安全软件开发中会遇到的问题和解决方案。

应掌握的知识要点：

- 安全的软件开发周期；
- 安全原则、规则及规章；
- 安全需求：攻击用例；
- 架构和设计评审/威胁建模；
- 安全的编码原则；
- 白盒/黑盒/灰盒测试；
- 判定可利用性；
- 安全地部署应用程序；
- 对安全漏洞进行管理；
- 软件安全编程。

## 2.1　安全的软件开发周期

在传统的软件开发生命周期(SDLC，Software Development Lifecycle)中，安全测试往往是一种事后反应，即安全检验和测试工作总是被拖延到软件开发完成之后。漏洞是软件的一种突变属性，它在整个设计和实现的周期中都会出现。因此，要求软件开发有事前、事中及事后的针对性方法。

在软件开发的过程中，缺陷发现得越早，修复费用就越低。因此，在整个生命周期中采用多个过程是非常重要的。

安全的软件开发在项目开始的时候就要启动，在许多软件开发团队中，安全测试阶段作为应用程序最终的“安全大门”来运行，通过这道大门，允许或阻止应用程序从安逸的软件工程环境进入到不安全的真实环境中。在软件开发生命周期的后期，随着引入这一过程，安全测试就背负了巨大责任：应用程序的安全以及团队的声誉，都依赖于它。

在软件已经实现之后才进行的安全测试，比如找外部团队来执行安全测试，然后提交一份测试报告，这种方式只是一种创可贴式的补缀解决方案。对于安全测试团队来说，一般只是把重点放在对软件安全的测试方法上，而几乎不考虑安全的软件开发周期所要求的围绕安

全测试的相关任务。然而，最有效的安全计划始于项目的开始阶段，要远早于任何程序代码编写工作，有效的安全过程在整个软件开发生命周期中都要使用。

### 2.1.1 将安全测试融入整个软件开发生命周期中

本节将讨论在软件开发生命周期的早期介入并解决安全问题的重要性。概要论述称作“安全的软件开发生命周期(SSDL，Secure Software Development Lifecycle)”的过程，其中包含了早期设置安全质量大门的有关内容。还将讨论为什么应该在软件开发生命周期中致力于解决安全需求问题，而且在初始阶段就应该开始这方面的工作。

在 SSDL 这一领域已经进行了大量研究，SSDL 描绘了一种结构化方法，用以贯彻和实现安全的软件开发。SSDL 方法体现了现代化快速应用程序开发研究计划带来的好处。这类研究计划在早期就开始考虑风险，贯穿每次软件构建的分析、设计和开发过程，这已经日益成为一种流行的做法。

遵守 SSDL，安全问题就可以在系统的生命周期早期得以评估和解决，包括业务分析期间、每次软件构建的需求分析阶段以及设计和开发阶段。这种早期介入方式，使安全团队能为安全需求规范、攻击用例以及软件设计提供质量审查。整个团队也将能对业务需求以及与其关联的风险有更全面的理解。最后，这个团队就能使用安全开发方法、威胁建模工作等设计和构建出最合适的系统环境，产生更安全的设计。

早期介入之所以重要，是因为需求和攻击用例组成了安全需求定义和衡量成功与否的基础和参考点。安全团队需要对系统或应用程序的功能规范进行评审。特别提出的是，至少要遵循下列准则对功能规范进行评估：

- 完备性：评估安全需求详尽定义的完备程度，并且当有可用的规章所定义的需求或安全策略时，还要验证安全需求是否与其相符。
- 一致性：确保每项需求都不会与其他需求相矛盾。
- 可行性：评估可行性程度，即评估需求能切实使用现有技术实现的程度，并且这种实现不能超出硬件规格、项目预算和项目计划进度以及项目人员的技能水平。
- 可测试性：评估可测试性，即评估测试方法能在多大程度上证明安全需求已经成功实现。
- 优先级：帮助每个人理解就风险承担而言的需求的相对值。应使用一种度量方式(如1～3)来规定优先级。如果某项需求对于系统安全性来说是至关重要的，就需要相应地指定其优先级。这种方法需要就这项需求综合考虑工程师的看法和与此相关的开销及技术风险。
- 规章：要求这种安全需求与本项目必须遵守的规章所规定的安全需求相符合。

在功能规范/需求定义阶段，应该确定安全测试策略。如果能时时考虑到系统安全的重要性，那么产品设计和编码标准就能提供合适的安全测试的工作平台。

SSDL 的目标就是确保安全的软件得以成功实现。它由 6 个主要部分组成：

- 安全原则、规则及规章；
- 安全需求；
- 架构、设计评审和威胁建模；
- 安全的编码原则；

- 白盒、黑盒、灰盒测试；
- 判定可利用性。

此外，还需要指定一个有关安全部署应用程序的过程。安全部署意味着软件安装时使用了安全的默认值，文件许可权限需要适当地进行设置，并在应用程序的配置中使用了安全设置。

软件在安全部署后，它的安全性就依靠在软件使用期持续地进行维护来保障，这就需要有一个周密的软件补丁的管理过程。此外，对出现的安全威胁要进行评估，并且需要对安全漏洞排定优先级并加以管理。

### 2.1.2　安全原则、规则及规章

安全原则、规则及规章在项目起始阶段就需要考虑，SSDL 的第 1 个阶段被视作保护性需求(Umbrella Requirement)。

在本阶段，可以基于特定的官方规章来定义一份系统范围的规范，在其中定义将应用到本系统的安全需求。“Sarbanes-Oxley Act of 2002(SOX)”就是这样一份公司范围的规章，它包含了特定的安全需求。例如 SOX 第 404 款这样规定：“必须使用各种内部控制来消除欺骗和滥用行为”。这可以作为基准来创建考虑到这种安全需求的公司范围安全策略。基于角色的许可权限等级、密码规格和控制以及访问等级控制就是为满足这一特定 SOX 条款的需求而需要实现并测试的内容。这里可以参考 1.3 一节中有关联软件安全标准的内容。

另一种公司范围的安全规章还可以定义成“本系统需要考虑并服从 HIPAA 隐私和安全规章”，或者“我们必须遵守金融现代化(The Graham-Leech-Bliley)条例”，或者“本系统需要满足支付卡行业数据安全标准(Payment Card Industry Data Security Standard)”，或者“本系统将满足 FISMA 标准”，这样来指出几条规定。有时会要求某个公司遵守各种标准，因而安全策略的创建者就需要考虑所有专用于这些标准的需求。

有些系统并不受任何规章条例或原则的影响。这种情况下，仍应开发安全策略。重要的是，不仅要以文档形式记录这些安全策略，还要通过对其进行跟踪和评估来使其成为一种不断发展的基本原则。

### 2.1.3　安全需求：攻击用例

安全需求是 SSDL 的第 2 个阶段。在各类需求文档中一种常见的错误是忽略了安全需求，以文档形式记录安全需求很重要，这不仅是因为安全需求有助于软件设计、实现以及测试用例的开发，而且能帮助确定技术选择和风险区域。

安全工程师应坚持要求将相关的安全需求与每项功能需求一起描述，并以文档形式记录下来。在每项功能需求描述中，都应该包含名为“安全需求”的一节，以在文档中记录所有特定功能的特有安全需求，这些安全需求有别于系统的安全策略或安全规范。

重要的是，这种用于需求开发和文档编写的原则必须在项目开始时就加以定义。例如，每个人都会同意类似“系统必须是高度安全的”这样的规定，但是，对于“高度安全”这种说法，每个人都有不同的解释。安全需求并不赋予系统任何特别的功能，而是限定或进一步定义了不应该允许系统以哪种方式处理某功能。这正是分析人员以攻击者的角度看待系统所应注意的地方。可以开发“攻击用例”来展现不允许的或未经授权的动作流。这些用例可以

帮助理解和分析前置条件(用例执行前系统必须所处的状态)及后置条件(用例执行后系统可能处于的一组状态)的隐含安全问题。用例的“包含(include)”关系可阐明许多保护机制，例如登录过程；用例的“扩展(extend)”关系可阐明许多检测机制，例如记录审计日志。攻击用例列出了系统可能被攻击的方式。

“安全缺点预防”是有助于在安全错误传播到后续开发阶段之前检测并规避这些安全错误的技术和过程的应用。缺点预防在需求阶段是最有效的，在该阶段，对所要求的内容进行更改来修正缺点，其影响程度较低。如果每个人都能在软件开发生命周期的开始就保持安全意识，他们就能帮助识别出遗漏点、矛盾点、模糊点以及其他可能影响项目安全性的问题。

“需求的可跟踪性”确保每项安全需求都以这样的方式来识别——这项安全需求可与系统中所有用到该需求的部分相关联。对于这项需求的每一次改动，都能识别系统中所有受这次改动影响的部分。

可跟踪性还使相关人员可以收集每项需求相关的信息，以及当某项需求发生变更时可能影响到的系统其他部分的信息，比如设计、编码、测试等。当有需求变更通告时，安全测试人员能够确保所有受影响的领域都得到相应调整。

安全需求示例：

- 该应用程序将存储敏感的用户信息，这些信息必须得到与 HIPAA 兼容标准的保护。为达到该目标，无论在哪里存储，都必须使用强加密来保护所有敏感的用户数据。
- 该应用程序将通过可能不受信任的或不安全的网络来传输敏感的用户信息。为保护这些数据，必须对通信信道进行加密，以防窃听；必须使用双向密码验证来防止代理攻击。
- 该应用程序必须对合法用户保持可用性。必须对远程用户的资源使用进行监视和限制，以防止或缓解拒绝服务攻击。
- 该应用程序支持具有不同权限等级的多用户使用，为用户分配了多种权限等级，并定义了每一权限等级所授权执行的操作。需要定义并测试各种不同的权限等级和绕过授权攻击的缓解措施。
- 该应用程序接受用户的输入，并将使用 SQL。需要定义 SQL 注入攻击的缓解措施。
- 必须对用户输入进行长度和字符有效性验证(必须定义合法的字符数据元素)。
- 该系统需要对各个用户及其身份鉴别保持跟踪。要求密码有关的涉密信息必须安全地存储。
- 该应用程序使用 C 或 C++编写。代码必须以这样的方式编写：始终跟踪并检查缓冲区长度；用户输入不能更改格式化字符串；整数值不允许溢出。若编译器支持栈探测方法，那就用这种方法。
- 该应用程序以 HTML 形式呈现用户生成的数据，因此必须具备 XSS 攻击的缓解措施。
- 该应用程序需要记录审计日志，要检验审计日志的安全性。
- 该应用程序将与其他受信的应用程序进行连接，必须对这些连接进行有效性检验并提供保护。
- 该应用程序将使用密码系统，且生成的数据必须使用安全的随机数生成器。
- 该应用程序将使用多线程或多进程，需要对其进行保护以防出现竞争状态。

- 本软件将打开文件并通常通过非授信链接来交换文件数据，例如通过 Internet 打开一个媒体文件，对所有从这个文件读取的数据不加信任，并对其进行有效性检验。
- 该应用程序需要安全部署，即软件需要使用安全的默认值来安装。需要设置适当的文件许可权限，并在应用程序配置中使用安全设置。

这仅是安全需求很少的一些示例，测试人员可以依据这些安全需求开发相应的安全测试用例。

### 2.1.4 架构、设计评审和威胁建模

架构和设计评审以及威胁建模代表了 SSDL 的第 3 个阶段。

安全从业者要充分熟悉产品的体系结构才能提出更好、更完备的安全策略、措施及技术方面的建议。安全团队及早介入可防止形成不安全的体系结构和安全性欠缺的设计，同样也有助于消除项目生命周期后期可能出现的应用程序行为的混乱。此外，及早介入还可使安全专家能够了解应用程度的哪些方面最关键，而从安全角度来看哪些要素风险最高。

这种认识使得测试人员能将重点放在那些最重要的部分，避免优先测试低风险部分，而将高风险部分放在次要位置。

软件开发过程的时间和资源都受到约束，在软件开发生命周期中，软件的上市时间是关键因素。在给定时间和资源不变的情况下，有必要排定测试项目的优先级，使得尽可能多地找出那些最严重的安全缺陷。“威胁建模”就是用来排定安全测试优先级的一种技术。通过威胁建模，在对应用程序的设计加以理解的基础上，可假定潜在的安全风险并对其进行评价。然后，根据攻击难易程度和攻击的影响严重性，将这些威胁进行分级并依次消除。这样，安全测试人员就可将注意力集中在那些攻击难度最低而影响最大的领域。

威胁建模的好处在于能发现的问题与代码评审及测试的不同，能发现较高层次的设计问题，而不是实现方面的错误。

通过威胁建模能够在编码实现为产品之前，及早发现安全问题。这有助于判断出应用程序的“风险最高”部分，这些都需要在整个安全开发工作中进行非常详细的审查。威胁建模另一个非常有价值的方面就是能让人有一种完备的感觉。在此说“本图中包含了所有的数据输入”，这是一种很有力的声明，在其他任何时候都不能做出这样的陈述。

### 2.1.5 安全编码原则

安全编码原则是 SSDL 的第 4 个阶段。

设计漏洞是一种设计中的缺陷，它使得程序不能安全地运行，而无论代码编写者如何完美地实现了这个程序都无济于事。实现漏洞是在实际软件代码编写中导致的安全性错误。

静态分析工具可通过扫描源代码或二进制的可执行文件检测出许多实现错误。这些工具对于找出类似缓冲区溢出这样的问题是非常有用的，其输出能帮助开发人员学习在开始的时候就对这些错误进行预防。

软件开发人员和测试人员应该参加有关培训课程，学习如何遵守这些安全的编码标准，从而开发安全的代码。

以安全的编码标准做基准，测试人员能够开发测试用例，检验是否遵循了这些标准。

还可以使用第三方提供的服务，将代码发送给他们，由他们来分析并找出其中的缺点。

使用第三方服务的好处是他们会从适应性因素或用户需求方面来验证代码的安全性。由于这只能在代码开发完毕后执行，因此检验要提出初始的标准并遵循。这样，第三方就可以有所侧重地验证这些标准的遵循情况，并将精力放在发现其他安全问题上。

### 2.1.6　白盒、灰盒与黑盒测试

白盒、灰盒与黑盒测试是 SSDL 的第 5 个阶段。

#### 1. 白盒测试

白盒测试常用于质量保证领域。有时也称作明盒测试、开盒测试或信息充分测试。在白盒测试中，对于测试人员来说，所有关于被测系统的信息都已知。在安全领域，这也可以认为是一种内部攻击。测试人员能够访问源代码和设计文档，这使得测试人员能够高效地工作。他们可以进行威胁建模或逐行审查代码，查找信息来指导测试数据的选择。

白盒测试是找出安全漏洞最有效的方法，更多的信息可使测试接口更快且完整地生成。这同样也提供了关于此系统安全性的精确印象，这是因为，这种印象不是依靠隐蔽式安全获得的，而隐蔽式安全是希望攻击者永远不会发现有关系统工作方式的信息。隐蔽式安全不是真正的安全，应该始终假定所有系统相关的信息最终都被发现或泄露出去。即使信息被泄露，设计和实现完善的系统依然会处于安全状态，这就是优秀的密码算法能公开发布而接受审查的原因，它们并不依靠保密来获得安全性。

进行白盒测试之前进行威胁建模，这可以揭示软件的攻击面，并将了解软件中设置了哪些功能来缓解这些攻击面所带来的风险。白盒测试要测试这些缓解措施。

缓解措施的例子：使用安全机制来应对会话标识符不够随机的问题。在通过威胁建模过程发现这项缓解措施后，安全测试人员可检查用来生成会话标识符的代码，然后自动执行创建新会话的过程，并记录此过程生成的这些会话标识符。然后，可对这些标识符进行数学分析，了解它们是否真正随机。

#### 2. 黑盒测试

黑盒测试是指以局外人的身份对系统进行分析，使用工具来检测系统的攻击面，并探查系统的内部信息。在没有系统内部知识时，测试人员要获得系统的情况。这时，信息泄露对于黑盒测试人员来说尤为重要，这是因为，相对于操作那些没有信息泄露的程序来说，这些泄露出来的信息有助于测试人员获得系统更多的情况。

许多测试人员都非常信赖黑盒测试技术，并用它作为白盒测试的补充。如果把规格说明书和设计文档看得过重，测试人员就可能遗漏系统中未被正确实现的或在文档中未提及的那些部分。这些规则之外的功能可能会存在安全缺陷，必须找出这些缺陷。黑盒测试可让测试人员探查所有攻击面，并为那些设计中没有的功能生成测试数据。一种常见错误就是在软件作为产品发布之前，没有去除那些仅用于调试的命令；另一种常见问题是在最后时刻抛出未在设计文档中正确记录的功能。黑盒测试可找出这些情况下存在的缺陷，而白盒测试可能不会注意这些问题。

当系统故意使用隐蔽式安全来保护信息时，这种方式常见于数字版权管理(DRM, Digital Rights Management)系统中，可以使用黑盒测试。纯软件的 DRM 不可能十分安全，因为攻击

者控制着 DRM 软件执行所在的系统。DRM 厂商能寄予希望的最佳方式就是把成功攻击的门槛抬到足够高，使攻击者放弃对其攻击。由技术精湛的反向工程团队执行的黑盒测试往往用来测试所用隐蔽式安全方法的强度。这通常需要专门的技术，已超出了质量保障团队的能力范围，因而进行这样的测试代价较高。当然，对于 DRM 系统，必须演示其使用的隐蔽方法的功效。

#### 3. 灰盒测试

理想情况下，在安全测试期间，会同时使用白盒和黑盒两种测试技术。白盒测试用于发现有文档说明的功能中的缺陷。当无法了解应用程序的内部信息时，使用黑盒测试来找出缺陷，这种组合称为灰盒测试。

应用程序的安全测试人员通常都会执行灰盒测试来找出软件漏洞，设计缺陷和功能的缺陷同等重要，都需要发现它们。由于安全测试人员可以使用源代码，因此应该利用这些代码来提高生产率。

将软件在调试器中运行而加以测试是一种理想方式，通过这种方式，可混合使用黑盒测试和源代码，从而使测试人员获得灰盒测试带来的好处。在 Windows 领域，可使用调试符合和源代码。微软开发人员套件(Microsoft Developer Studio)是一种典型的调试器，它使得测试人员可以轻松地在栈和内存之间巡查，对诸如类和结构这样的复杂变量进行考察。在 UNIX 领域，通常使用 gdb 执行这类工作。

在软件运行于调试器中时，可引入常见的黑盒测试工具来用于执行中的程序，例如侦探程序(Fuzzer)和自动化的回归测试套件。测试人员可在危险的代码行处设置断点，从而查看这些代码是否受程序外部输入的影响。这些危险的代码行可通过代码审查、简单地使用字符串查找(grep)或搜索代码来找出。

危险代码的一个例子就是 C 语言在格式化字符串中有%s 的 sprintf 语句。如果复制到目标缓冲区的源缓冲区太大，就会出现缓冲区溢出条件。但并不是有这种条件的 sprintf 语句都可以被发现并利用。灰盒测试能找出那些真正可被利用的代码行。

如果在代码开发期间发现了问题，就要立即修补潜在可利用性的代码。但是，出于对打补丁开销的考虑，许多开发团队并不喜欢修补那些已经发布的或者已经投入生产的代码的安全缺陷。用于发现漏洞的灰盒方法为开发团队提供了关键的可利用性信息，这可以帮助他们就是否修补代码中的潜在漏洞做出决定。

### 2.1.7　判定可利用性

判定可利用性是 SSDL 的第 6 个阶段。

理想情况下，在 SSDL 的测试阶段发现的每个漏洞都容易修正。然而，依漏洞的产生原因不同(是设计错误还是实现错误)，解决这个问题所需的工作量差别会很大。在计量其带来的风险时，漏洞的可利用性是一个重要因素。可使用此信息对漏洞的修补和其他开发需求(如实现新的功能特性和解决其他安全问题)进行优先级排序。

判定漏洞的可利用性包含对以下 5 个因素的权衡：

- 攻击者企图探测并利用这个漏洞所需的访问权限和定位技术；
- 成功利用此漏洞能获得的访问级别或权限；

- 探测并利用此漏洞的时间和工作量因素；
- 探测和利用潜在的可靠性；
- 探测和利用尝试行为的可重复性。

### 1. 时间

某些漏洞可能需要相当长的时间来探测并利用。这可能表示漏洞利用本身需要传输大量的数据、执行大量重复的尝试或使用相当数量的计算资源。这类需求的常见例子包括发送数GB的通信流、暴力攻击内存地址或破解密钥。如果加密漏洞需要几千年的时间来计算才能利用的话，这就表示这种风险非常低。

### 2. 可靠性和再现性

漏洞的严重程度取决于漏洞可被攻击者利用的可靠性和再现性如何。可靠性和再现性与漏洞是低级别的还是高级别的有关。

低级别漏洞通常会破坏运行中的应用程序或编程语言运行时程序的状态。例如，缓冲区溢出是一种常见的低级别漏洞，既会破坏应用程序的运行状态，也会破坏编程语言运行时的状态。出于这种原因，这类漏洞并不总是百分之百地可再现或者说百分之百可靠。这些漏洞可能会导致应用程序崩溃，也可能需要依赖应用程序或进程处于某种状态时才可能导致应用程序崩溃，而这种状态对于攻击者来说是无法预测或控制的。

高级别漏洞通常涉及应用程序的逻辑错误。常见的高级别漏洞包括：SQL注入、跨站执行脚本以及其他由程序逻辑错误引起的漏洞。这些漏洞不倾向于导致应用程序崩溃，一般都非常可靠且可再现。高级别漏洞往往都很简单，而且其可利用性是完全确定的。像Java、PHP、C和C++这样的语言中存在的高级别漏洞通常总是可以可靠地利用。

通常情况下，高级别漏洞(如目录遍历、SQL注入以及跨站执行脚本)的可靠性和再现性高，而复杂漏洞(例如破坏程序状态的缓冲区溢出)的可靠性各不相同。这很大程度上取决于漏洞本身，还取决于构建一次漏洞利用的技能和投入的时间。精确测量某个漏洞利用的可再现性会比较困难。然而，一条基本的准则就是栈和数据段溢出的漏洞利用倾向于具有高可再现性，反之，堆溢出漏洞利用的可再现性就较低。重要的是，要注意应用程序是否会重新启动自己，这是一种重要的因素。如果攻击者可以多次尝试漏洞利用，堆溢出漏洞利用的可靠性和可再现性可能变得相当高。

### 3. 访问

利用漏洞通常能为攻击者提供比之前更多的访问权。通过漏洞得到的这种访问授权，其形式可能是用户权限、网络访问或对其他保护资源的访问权。在一个简单案例中，漏洞可能允许攻击者直接从Web服务器的文件系统中读取文件，包括那些位于这个Web服务器根文档目录之外的文件(即“目录遍历”)，或者如同PHP和JSP页面这样的服务器端解释的源代码文件(即“源代码泄露”)。这种漏洞会为攻击者提供更多信息，攻击者可使用这些信息来攻击系统，不过，利用漏洞并不会得到完全的系统访问权。然而，如果远程攻击者可利用漏洞来获得UNIX影子密码文件，并将其破解而获得有效的系统身份信息，那么漏洞可导致攻击者获得系统访问权。

另外一系列漏洞是那些导致“命令执行”或“任意代码执行”的漏洞。命令行漏洞可能

会出现在这样的 Web 应用程序中：程序未能正确地对输入的数据进行无害处理，而将这些输入用于某个命令行程序中，使得攻击者能在其中注入自己的命令并得以执行。任意代码执行漏洞包括缓冲区溢出漏洞和格式化字符串漏洞，这类漏洞允许远程攻击者将应用程序的执行重定向到自己提供的机器代码。这将导致攻击者获得正在运行的程序的全部权限，这种情况下，应用程序可能拥有超级用户或 Administrator 权限，或者没有任何权限而运行；或者先前的权限被撤消，转而运行于一种较低权限的状态。然而，有时权限并没有正确地或完全地撤消，并可能在攻击者控制这个应用程序之后就立即恢复。

其他常见漏洞，如 SQL 注入和跨站执行脚本，将导致各种不同的访问授权等级。一个 Web 应用程序中存在的 SQL 注入漏洞可能会使得远程攻击者获得该应用程序数据库的访问权限。在得到良好安全防护的部署中，这可能是一个数据库上无特权的账户，这个账户只有对该应用程序数据的访问权。然而对于攻击者来说，可能这就足够了，因为有了这种权限，就能读取或修改该应用程序的任何数据。某些数据库配置可能还允许更多的访问权限。

跨站执行脚本漏洞使得攻击者可在存在漏洞的Web 站点的安全环境下，将可执行脚本代码或其他任何 HTML 代码注入到受害用户的 Web 浏览器中。根据该 Web 站点所能完成任务的差别，攻击者可获得各种不同的访问权限级别。

**4. 定位**

要利用一个漏洞，攻击者必须能与这个存在漏洞的应用程序进行交互，并能访问到含有该漏洞的代码。是否可以通过 Internet 进行攻击由这个应用程序是不是网络软件以及使用哪种协议进行通信决定。某些应用程序使用像 IP 组播(Multicast)这样的协议，而这种协议典型的情况下是不能被路由器转发到其他子网的。还有许多应用程序采用了自定义协议，而防火墙可能不允许这种协议穿过。此外，某些应用程序使用了 HTTP 这样常见的 Internet 协议，因而这也就使得攻击可以穿过 Internet 防火墙得以执行。一般来讲，可通过 Internet 进行攻击时风险最高，因为这种攻击可在任何位置发起。

一旦攻击者能够访问到存在漏洞的应用程序，他必定也能访问到存在漏洞的代码。这里重要的屏障就是鉴别身份信息以及其他目标特有的知识。如果这个漏洞在登录系统之前就会出现的话，比起要求使用管理身份来利用的漏洞来说，的确造成了较高风险。此外，攻击者可能需要某些其他目标特有的知识，比如该应用程序正在监听的动态选择的 TCP 端口、Web 应用程序中的某个站点特有的路径或其他任何完成协议协商需要的参数。例如，某个自定义协议可能要求提前配置好某些加密参数，并可能不加通告地舍弃那些未使用的正确算法、密钥、密钥长度或初始化向量等。

最后，了解攻击是“主动的”还是“被动的”也很重要。攻击者可对已知目标发起攻击，也可以等待某个用户的特定活动之后再采取行动。例如，大多数服务器软件中的漏洞都允许主动攻击。攻击者可在任何时间发起攻击。与此相对的是被动攻击，比如针对某个 Web 浏览器漏洞进行攻击。攻击者必须强制某个访问某个 Web 服务器的用户处于控制之下，才能利用这个漏洞。通常来讲，允许主动攻击的漏洞比被动攻击的漏洞有更大的风险。

每个漏洞的风险是解决各个漏洞之间以及与其他开发任务之间排定优先顺序的依据。可利用性需要周期性重新评估，这是因为，随着时间的流逝，漏洞的可利用性总是变得更加容易，例如密码机制变得比以前更脆弱，人们也推出了新的加密技术。

### 2.1.8　安全地部署应用程序

计算机软件的部署指计算机软件投入使用过程中的所有相关环节，部署软件系统涉及从计算机软件源头(可以理解为服务器端)将软件组成部分传递或复制到客户端，一旦软件被部署，客户就能使用软件系统。

安全地部署和维护应用程序的过程应放在生命周期的最后。当然，需要在开始时就为使应用程序能被安全地部署而进行设计。安全部署意味着软件安装时使用了安全的默认值，文件许可权限经过了适当设置，并在应用程序配置中使用了安全设置。

软件部署的结果对软件系统运行时的性能，如反馈时长、系统稳定性、输入输出的数据量、易用性、安全指标等，有很重要的影响，而且这种影响作用对分布式软件系统更加明显。因为在分布式软件系统中，软件系统往往需要根据分布式平台所运行的应用不断地微调、迭代、运维、扩充软件的设备设置信息，从而达到优化分布式系统工作效率的目标。分布式软件系统的编译、唤醒、反唤醒、迭代、环境适应、反向提出、删除等活动可以看成分布式软件系统的全局部署过程，随用户需求和分布式运行环境的变化而变化。近年来，如何实现软件系统的优化部署是软件部署和分布式系统研究的重要内容。

随着软件系统复杂性和软件运行环境复杂性的不断增加，软件部署在软件生命周期中变得越来越重要。尽管软件部署受到越来越多研究人员的关注，但由于软件自身复杂性和软件部署环境复杂性的增加等因素，软件部署目前还存在以下几个方面的问题：

1) 软件规模庞大。随着科学技术的进步，计算机软件系统的规模越来越大，复杂性迅速攀升。计算机软件系统常由众多对象或组件构成，多个对象或组件之间存在相互依赖关系，这些依赖关系直接影响软件部署过程中的活动。

2) 工作环境多样性。随着网络技术的快速发展，计算机软件系统的部署环境由原来独立、非动态、统一标准的环境转变为开放、非静态、多种构架的环境，这种开放、非静态、多种构架的网络环境给计算机软件的部署带来了难度。例如，软件部署不仅需要考虑对数据文件的需求状态，还要考虑组件之间的数据信息等传递情况并兼顾服务器的工作环境等，提升了计算机软件部署的复杂度。

3) 自调整问题。软件的不断发展需要软件部署能够进行更迭操作，为了满足用户功能性方面的需求变化和计算机工作环境的变化，软件部署需要进行自我调整。因此，完整的部署过程是软件自我调整的过程。

4) 适应问题。软件部署活动之间是有关联的，一个活动的发生可能需要其他活动的发生为前提，也可能导致其他活动的发生。例如，在服务器端组件进行更新或再配置之前，必须首先反激活所有客户端的组件，以免影响正常的交易。因此，适应问题是软件部署中的一个重要问题。

5) 安全问题。互联网的发展非常迅速，软件的部署也有了更多的数据机密性、访问控制和统一规范性等安全方面的问题。例如，计算机软件在编译执行和版本更迭中需要一定的机密性，这需要网络传输协议的内容不被非法窃取。另外，由于不同的分支或组织具有不同的权限，软件部署还要解决身份验证的问题。最后，由于软件系统部署在多台服务器中，从而可以保证资源文件信息的完整性以避免资源文件的破坏或丢失情况发生。

此外，必须对安全的部署进行经常性监控，同时，必须对漏洞进行管理。

### 2.1.9　角色和职责

安全到底是谁的职责，这个问题往往并不是那么清晰。为有效开展安全测试工作，就必须澄清角色和职责问题。在安全的软件开发生命周期中，安全是许多人的职责。不能将安全仅依赖于网络小组安装入侵检测系统和防火墙，以及运行一些用于安全的网络工具。重要的是要对角色和职责进行定义，这样，每个人都知道由谁测试什么内容，举例来讲，这样就可以使得应用程序测试团队不再假定网络测试工具也会捕捉到应用程序的安全漏洞。

项目经理或产品经理应该编写安全策略。如果可能的话，这些安全策略可以基于指定的相关标准来编写。如果没有专门的安全角色来负责安全认证的话，产品经理或项目经理还要负责处理安全认证过程的工作。架构师和开发人员负责提供设计和实现细节、确定和研究安全威胁以及执行代码评审工作。测试人员驱动系统的关键分析、参加安全建模工作、确定和研究安全威胁并构建白盒和黑盒测试。项目经理管理计划进度，并控制每份文档及其日期。安全过程经理可检查指导安全建模、安全评估以及安全编码培训。

## 2.2　软件安全编程

面对越来越频繁的软件安全隐患带来的损失，对软件的开发者——软件工程师，提出了更高的要求，要求程序员能编写出错误更少的程序，并能及时修复软件中出现的突发问题，切实为软件使用者服务。本节讲解的安全编程技术旨在解决这些问题。安全编程是软件质量的重要保证，在软件开发和程序设计中占有重要地位。

不过，实际的软件工程中，安全隐患的出现往往来源于多个方面，给软件系统带来的危害也有很多方面。安全问题的出现，由于原因众多，而某些安全问题又具有不间断发生、难于调试等特点，很难用单纯的理论来全面阐述安全编程问题。基于这个考虑，安全编程的内容只能针对各个侧面来进行阐述，如异常情况下的安全、线程操作中的安全、数据安全加密等。

### 2.2.1　内存安全

内存安全关系到整个程序的安全，在软件开发和程序设计中占有重要地位。如果程序员稍微疏忽，很容易出现安全隐患。

内存数据涵盖了很多方面，如字符串、数组、整数都在内存中以不同形式存在，它们在被操作的过程中，具有哪些特点，什么样的操作能够产生攻击，这些都是程序员编程时需要重视的问题。

内存安全主要是缓冲区溢出问题。该问题在字符串复制或其他函数使用时很容易出现，处理不当，会给程序留下安全漏洞，成为攻击的目标；其次是整数溢出问题，整数由于其保存的特殊性，某些特殊的计算可能产生奇怪的结果，如果处理不当，照样会成为隐患；另外，数组越界、字符串格式化都是需要重点考虑的问题。

缓冲区溢出攻击，由于实现起来比较方便，已经成为一种比较常见的安全攻击手段。因此，相对于其他漏洞，缓冲区溢出漏洞比较普遍。

攻击者可以通过很多手段利用缓冲区溢出漏洞并且进行攻击。一般来说，利用缓冲区溢出攻击的目的是使攻击者取得某些程序的控制权，执行某种特权功能，实现非法操作；极端情况下，如果该程序具有管理员权限，那就相当于控制了整个主机。

攻击者为了达到目的，一般情况下，攻击行为分为两步进行。

第一步：在程序的地址空间放入一些攻击性数据，故意让缓冲区溢出。该方法中一般有两种攻击方式：

1) 直接输入法。攻击者向被攻击的程序输入一个字符串，让缓冲区溢出，程序会把这个字符串放到缓冲区里。而该字符串中包含某个指令序列，攻击者因而猜测出可以攻击的漏洞地址。

2) 传递参数法。这种情况下，攻击者想要执行的代码存在于漏洞程序中，只需传递一些参数就可以让它运行；例如，攻击代码要求执行 exec(“某个命令”)，而在被攻击程序的库中有一个函数为 exec(arg)，那么攻击者只需将命令参数传给被攻击程序。

第二步：精心设计溢出的数据，让程序执行攻击者预想的功能，也就是改变程序的执行流程，跳转到攻击者安排的攻击代码。

让程序跳转到相应的程序代码，一般情况下有如下方法：

- 利用另一个函数的返回地址。函数调用时，堆栈中会留下函数结束时返回的地址，指示函数结束后会执行的功能。攻击者可通过缓冲区溢出，改变程序的返回地址，使返回地址为攻击代码。
- 直接利用函数指针。由于函数指针可用来定位函数的位置，攻击者只需在函数指针附近将缓冲区溢出，用攻击函数的指针来覆盖原函数指针，达到攻击目的。

解决缓冲区溢出的方法有如下几种：

- 积极检查边界。由于 C 和 C++允许任意的缓冲区溢出，没有任何的缓冲区溢出边界检测机制来进行限制，因此一般情况下，所有开发者需要手动在自己的代码中添加边界检测机制。不过，也有一些优化技术来减少手工检查的次数，如使用 Richard Jones 和 Paul Kelly 开发的 gcc 补丁、利用 Compaq 的 C 编译器等。
- 不让攻击者执行缓冲区内的命令。这种方法使得攻击者即使在被攻击者的缓冲区中植入了执行代码后，也无法执行被植入的代码。
- 编写风格良好的代码。养成习惯，不要因为一味追求程序性能，而编写一些安全隐患较多的代码，特别是不要使用一些可能存在漏洞的 API，减少漏洞发生的可能。可用一些查错工具，限制一些可能具有缓冲区溢出漏洞攻击的函数的调用(如 strcpy 和 sprintf 等)。
- 程序指针检查。程序指针检查不同于边界检查，程序指针检查是一旦修改了程序指针，就会被检测到，被改变的指针将不被使用。这样，即使攻击者成功地改变了程序的指针，因为系统事先检测到了指针的改变，这个指针也将不会被使用，达不到攻击的目的。

### 2.2.2　进程与线程安全

进程和线程是两个范围不同的概念。进程是程序在计算机上的一次执行活动，是操作系统进行资源分配的单位，通俗地讲，是正在执行的程序。

线程是进程中的一个实体，是被系统独立调度和分派的基本单位，它可与同属一个进程

的其他线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。实际上，多线程最直观的说法是：让应用程序看起来好像同时能做好几件事情。在实际应用开发过程中，经常会出现一个程序看起来同时做好几件事情的情况，如：

- 程序进行一个用时较长的计算，希望该计算进行时，程序还可以做其他事情；
- 软件要能够接受多个客户的请求，而让客户感觉不出等待；
- 媒体播放器在播放歌曲的同时也能下载电影；
- 财务软件在后台进行财务汇总的同时还能接受终端的请求等。

这些情况下，多线程就能够起到巨大作用。

线程与进程的开发和相关操作，在程序设计中具有重要地位，线程与进程的安全和系统的安全息息相关。

一般说来，线程的安全性问题主要来源于其运行的并发性和对资源的共享性；进程的安全性问题主要在应用级别，在于其对系统的威胁性，不过对于系统软件的开发者，进程安全的考虑需要更加深入。

每个程序至少自动拥有一个线程，称为主线程。当程序加载到内存时，启动主线程。这是由于多线程的机制实际上相当于 CPU 交替分配给不同的代码段来运行：也就是说，某一个时间片，某线程运行；下一个时间片，另一个线程运行；各个线程都有抢占 CPU 的权利，由操作系统决定哪个线程进行抢占。由于时间片的轮转非常快，用户感觉不出各个线程抢占 CPU 的过程，看起来好像计算机在“同时”做好几件事情。

一个线程从创建、运行到消亡的过程，称为线程的生命周期。用线程的状态(state)表明线程处在生命周期的哪个阶段。线程有创建、可运行、运行中、阻塞、死亡 5 种状态。通过线程的控制与调度可使线程在这几种状态间转换。这 5 种状态详细描述如下：

- 创建状态：使用 new 运算符创建一个线程后，该线程仅是一个空对象，系统没有分配资源；
- 可运行状态：使用 start( )方法启动一个线程后，系统分配了资源，使该线程处于可运行状态(Runnable)；
- 运行中状态：占有 CPU，执行线程的 run( )方法；
- 阻塞状态：运行的线程因某种原因停止继续运行；
- 死亡状态：线程结束。

线程的安全隐患可能出现在各个状态。一般来说，线程的安全性来源于两个方面：多个线程之间可能会共享进程的内存资源；CPU 的某个时间片分配给哪个线程使用，默认情况下无法由用户控制。

默认情况下，线程都是独立的，而且异步执行，线程中包含了运行时所需的数据或方法，而不需要外部资源或方法，也不必关心其他线程的状态或行为。但是多个线程运行时在共享数据的情况下，就需考虑其他线程的状态和行为，否则就不能保证程序的运行结果的正确性。在某些项目中，经常会出现线程同步的问题，即多个线程在访问同一资源时，会出现安全问题。

所谓同步，就是在发出一个功能调用时，在得到结果之前该调用就不返回，同时其他线程也不能调用这个方法。通俗地讲，一个线程能否抢占 CPU，必须考虑另一个线程中的某种

条件，而不能随便让操作系统按照默认方式分配 CPU，如果条件不具备，就应该等待另一个线程运行，直到条件具备。

有些情况下，多个线程合作完成一件事情，此时线程之间实现了协作。如一个工作需要若干个步骤，各个步骤都比较耗时，不能因为它们的运行，影响程序的运行效果，最好将各步用线程实现。但由于线程随时都有可能抢占 CPU，可能在前面一个步骤没有完成时，后面的步骤就已经运行，该安全隐患造成系统得不到正确结果。

死锁(Dead Lock)是指两个或两个以上的线程在执行过程中，因争夺资源而造成的一种互相等待的现象。此时称系统处于死锁状态，这些永远在互相等待的线程称为死锁线程。

产生死锁的四 4 个必要条件是：

- 互斥条件，资源每次只能被一个线程使用；
- 请求与保持条件，一个进程因请求资源而阻塞时，对已获得的资源保持不放；
- 不剥夺条件，进程已获得的资源，在未使用完之前，不管其是否阻塞，无法强行剥夺；
- 循环等待条件，若干进程之间互相等待，形成一种头尾相接的循环等待资源关系。

这 4 个条件是死锁的必要条件，只要系统发生死锁，这些条件必然成立，而只要上述条件中的一个不满足，就不会发生死锁。

线程控制主要是对线程生命周期的一些操作，如暂停、继续、消亡等。Java 提供了对线程生命周期进行控制的一些函数：

- stop( )，停止线程；
- suspend( )，暂停线程的运行；
- resume( )，继续线程的运行；
- destroy( )，让线程销毁。

线程生命周期中的安全问题主要体现在：线程暂停或终止时，可能对某些资源的锁并没有释放，它所保持的任何资源都会保持锁定状态；线程暂停后，我们无法预计它什么时候会继续(一般与用户操作有关)，如果对某个资源的锁长期被保持，其他线程在任何时候都无法再次访问该资源，极可能造成死锁。针对这个问题，为减少出现死锁的可能，Java 1.2 中将 Thread 类的 stop( )、suspend( )、resume( )以及 destroy( )方法定义为“已过时”方法，不再推荐使用。

进程是执行中的程序，对每个进程来说，都有自己独立的一片内存空间和一组系统资源。进程由进程控制块、程序段、数据段三部分组成。在进程概念中，每个进程的内部数据和状态都是完全独立的。

一个进程可以包含若干线程，进程也有运行、阻塞、就绪三种状态，并随着一定条件而相互转换。

由于进程的独立性，从应用角度看，进程安全比线程安全更受重视，一般针对已有的进程进行安全方面的控制。比如：在系统安全中发现并清除病毒进程，在网络应用中优化守护进程或端口扫描进程等。

从开发者(编程)角度看，进程的安全所需要考虑的问题与线程类似，但由于线程能够共享进程的资源，所以线程安全一般考虑的问题比进程安全要多。不过，对于开发多个进程能够运行的系统软件(如操作系统)，进程的安全就应该重点考虑了。一般情况下，此时考虑的问题和线程安全类似，因为在这种软件中，各个进程在使用系统中有限的资源，与线程安全

中考虑的问题相似。

### 2.2.3　异常与错误处理中的安全

异常/错误处理是程序设计中的常见内容，异常/错误处理的技巧和程序的安全性具有密切的关系。科学的异常/错误处理方法是系统安全性的重要保障。

一般来说，开发过程中可能出现的问题有如下几种：

编译错误，语法写错了，比如在 C++中，int a 写成了 Int a，这种错误，编译器能够进行提示，一般比较容易解决。

运行错误，语法没有问题，但在运行时发生了问题。比如连接数据库的代码本来是正确的，但运行时数据库突然断电，程序不能正常运行，这是在代码编写阶段应该预计到的，可由异常处理解决(Java 语言中定义了 Error 和 Exception，都是为了解决此类问题)；在某些语言(如 VB)中，没有面向对象的异常处理机制，此时设计了面向过程的错误处理方法来解决这个问题。

另一种是逻辑错误，程序语法没有问题，也没有异常，但得不到正确结果，这需要靠程序员非常高超的编程经验来进行处理，这不属于本节探讨的范围。

如前所述，异常主要是针对程序语法没有问题时，在运行的过程中出现的突发情况。异常的出现，是在程序编译通过的情况下，程序运行过程中出现一些突发情况造成的，处理这些突发情况，需要有良好的预见性，预先进行处理，以保证系统的安全性，这对程序员提出了更高要求。实际上，不可能预见程序可能出现的所有异常。

常见异常可能出现的场合如下：

- 访问数据库时，数据库停止工作；
- 访问文件，文件恰好被另一个程序访问；
- 输入一个以 0 作为除数的数值；
- 类型转换、对象未分配内存等。

从上面可能出现异常的场合可以看出，异常是几乎所有高级语言都可能出现的情况，在面向对象的语言中，C++、C#等也会出现类似的情况，包括一些非面向对象的语言，如 VB，也必须面对程序运行过程中的异常现象。虽然处理方法不同，但本质类似。

当系统底层出现异常时，实际上是将异常用对象封装起来，传给调用方(客户端)，俗称抛出(throw)。如在程序里面发生了数字格式异常，这个异常在底层就被封装成 java.lang.NumberFormatException 对象抛出。异常对象抛出给函数的调用者，如果调用者具有对异常处理的代码，则对异常进行处理；否则将异常继续向前抛出；如果直到用户端还没有对异常进行处理，异常将在标准输出(如控制台)中打印。对于非面向对象语言，异常出现的原理类似。

程序中可能出现的异常有很多种类，如算术异常(除数为 0 等)、数组越界异常、类型转换异常、分配内存异常、数字格式异常等。

异常出现后，可通过查看文档来了解发生的原因。但是，了解异常出现的原因并不是最终目的，为保证系统的正常和安全运行，将异常进行有效的处理，才是我们所需要的。要进行异常处理，首先必须对异常进行捕获(catch)，在面向对象的语言中，可以有两种方法捕获异常：就地捕捉异常和将异常向前端(调用方)抛出。当一个模块中可能出现异常时，一般情

况下，可就地捕捉异常，过程如下：

- 用 try 块将可能出现异常的代码包起来；
- 用 catch 块来捕获异常并处理异常；
- 如果有一些工作是不管异常是否出现都要执行的，则将相应的代码用 finally 块包起来。

如前所述，一个 try 后面必须至少接一个 catch，可以不接 finally，但最多只能有一个 finally。我们知道，代码中可能出现的异常有很多种类。如 Java 中常见的就有：未分配内存异常、未找到文件异常、数据库异常、格式转换异常、类型转换异常等。由于无法对所有异常进行预见，怎样尽可能多地捕获程序中可能出现的异常呢？

由于 try 块后面可以接多个 catch 块，因此可用某一个 catch 捕获某种异常。当 try 中出现异常时，程序将在 catch 中寻找是否有相应的异常类型的处理代码，如果找到就处理，如果没有找到就继续向下找。

在异常处理过程中，finally 块是可选的，实际上，finally 是为了更大程度上保证程序的安全性。异常通常有两种处理方法：就地处理和向客户端传递。就地处理就是在出现异常的模块中处理异常。程序中的异常，是就地处理比较好还是向客户端传递比较好？此处要遵循下列原则：就地处理方法可以很方便地定义提示信息，对于一些比较简单的异常处理，可以选用这种方法；向客户端传递的方法，其优势在于可以充分发挥客户端的能力，如果异常的处理依赖于客户端，或者某些处理过程在本地无法完成，就必须向客户端传递。例如，数据库连接代码可能出现异常，但是异常的处理最好传递给客户端，因为客户端在调用这块代码时，可能要根据实际情况，获取环境参数，进行比较复杂的处理。

这样做的好处是：在客户端可以进行更丰富的异常处理，不仅增加了可扩展性，也可以做到更安全的代码保障。所以，一般情况下，模块中的异常，如果确定可以就地处理则就地处理，否则就应该向客户端抛出。不过，异常不断向客户端抛出，会增加系统开销。实际上，在自定义异常的时候也会遇见相同的问题，其原则类似。

综合各种语言的特性，异常处理机制一共有两种：

1) 面向对象的异常处理机制。主要针对面向对象的语言，一般使用 try-catch-finally 结构来处理异常，前面所述的异常处理机制都是面向对象的异常处理机制。

2) 面向过程的异常处理机制。实际上，对于一些非面向对象的语言，如 VB，早期也具有异常处理机制，这就是面向过程的异常处理机制。甚至在面向对象的语言，如 VB.NET 中，除了推出面向对象的异常处理机制外，也保留了面向过程的异常处理机制，主要以 On Error 结构为代表。

### 2.2.4 输入安全

输入是一个很广泛的概念，既是用户和软件之间的交互手段，也是软件内部模块之间的交互手段。针对软件用户的输入有很多类型，比如：用户在软件上输入一个命令，进行相应操作；用户输入自己的账号密码，进行登录验证；用户输入一个关键字，进行查询等。模块之间进行数据传递时，也会有相应输入，比如：一个模块调用另一个模块时，输入一些参数；一个模块读取一个配置文件，以对自己的行为进行配置等。从程序本身的角度讲，很多情况下，软件的安全问题就出在输入；从攻击者的角度讲，输入是进行攻击的重要手段。经过调

查总结，大部分的软件安全问题来源于应用程序接受输入数据前，没有进行安全性验证。很明显，如果任由用户输入而不进行检查，用户就可以输入其他对系统有害的命令，如 rm(删除)命令，带来的危害是巨大的。解决上面问题的方法显然是进行安全性验证，对于编程人员来说，程序的所有输入数据，在被进行安全性验证之前，都被认为是有害的。一旦忽略了这条规则，程序就可能遭受攻击。

以上规则看似容易，也容易理解，但在传统情况下，安全性验证往往被忽略掉。主要原因如下：

1) 在同一软件中，由于每一个输入到达最后的执行模块的过程中，都需要经过许多关口，每个关口都有可能进行检查。但就是因为这样，许多开发人员都假定这些数据在通过其他关口时，已经由其他关口的应用程序函数检查过了，回避了对输入的检查，不愿意牺牲性能去对数据进行多次校验，结果导致大家都没有进行验证。

2) 随着软件的分工，现在许多应用程序的功能都分块分布在不同机器上(如客户机器和服务器上，或者对等机器上)，开发人员有充足理由依赖应用程序的其他模块提供安全检验。

从上面的例子又可以看出，输入安全解决不好，在严重的情况下，可能会带来巨大的危害。要想防御应用程序可能受到的输入攻击，最简单有效的方法是：在对输入进行任何一步处理之前，必须对数据安全进行验证。数据安全验证，说起来比较容易，做起来要考虑很多问题，由于很多软件开发者感觉数据安全验证太容易，反而会忽略科学的方法。

在对输入进行检查时，为了保证实际工作的可行性，在设计时，可以采用以下几个策略：

1) 尽量让程序可以输入的入口少一些。这样的话，如果程序分为若干个模块，那么攻击者直接和某些模块通信的概率大大降低，也就是说，攻击者进入程序的途径将大大减少，同时限制了各模块之间的通信路径进行攻击的可能，安全验证的代价大大减小。

2) 尽量让输入所允许的输入类型少一些。这样可以让验证的工作更加简化。比如，如果仅允许输入的值为数字，那么验证时只需针对数字进行验证，验证是相对简单的；如果将输入设计为任何字符串都可以输入，那么将要考虑更多问题，验证难度将会增加很多。

3) 严格检查不可信的输入。不仅在数据最初进入程序时要执行检查，而且在程序实际使用这些数据时，也要进行检查。当然，检查的项目可以不一样，但是检查应该是时时存在的。不过，相对来说，更重要的是数据在使用之前的检查。一般情况下，我们可以采用如下方法：数据在进入模块时，在各个模块内进行针对该模块的安全检查。

4) 转变观念，从定义“非法”到定义“合法”。安全程序开发人员往往有个误区，它们首先定义的是“什么样的数据非法？”，这个定义很容易给出，比如，email 地址中可以定义没有“@”符号为非法，但这是不安全的。因为不可能将所有非法的数据都加以定义，攻击者非常聪明，他们常常会想出其他的非法数据。定义“什么是非法”，容易想到，但是无法定义全；但是定义“什么是合法”，就相对容易得多。“正确答案只有几个，而错误答案可以有很多个”，就是这个道理。

所以应该做的是确定“什么样的数据是合法的？”。在数据输入时，检查数据是否符合定义，只接受符合定义的数据；而不是检查数据是否不符合定义，拒绝不符合定义的数据。

例如，有一个程序，根据用户的输入，创建一个文件。很显然，有些字符，如“/”，是不允许的。但是仅去检查一个字符也不够，其他字符，如控制字符、空格、横线等，都有可能不合法。即使创建了一个“非法”字符的列表，也可能没有办法定义完全，因为总可能有

没有考虑到的情况。因此，正确的方法应该是：确定文件名输入的一种安全的特定格式，而拒绝不符合这种特定格式的所有输入。

### 1. 数字安全问题

数字的输入安全是比较常见的。比如在表单上输入一个人的年龄，一般就会有范围限制。对数字的安全检查主要有：格式，如整数、小数等；精度，如小数保留的位数等；范围，如某个输入数字的大小取值范围等；类型和范围的匹配，如为了预防整数溢出，对短整数的输入进行范围检查。

针对数字输入的安全要注意以下几点：

1) 确定数字格式。比如，有的系统中数字是阿拉伯数字，也有的系统支持中文数字(如一二三，甚至壹贰叁等)输入，有的系统中数字每三位就有一个逗号等。一般情况下，可用正则表达式来验证字符串是不是数字，然后进行数字的安全检查。

2) 对于负数的验证。一般最好不要根据有没有符号位来确定该数是不是负数。因为有些程序中，如果输入一个很大的正数，也可能导致数值“溢出”而变成一个负数，对于负数的验证要进行一些底层的判断。

3) 特别要注意判断数值溢出(如整数溢出)的问题。

### 2. 字符串安全问题

关于对字符串的验证，有以下问题值得注意：

1) 如果使用正则表达式，最好明确指出要匹配数据的开始(通常用“^”来标识)和结束(通常用“$”来标识)。否则，攻击者可能在输入中嵌入攻击文本，并能绕过安全检查。

2) 尽可能在输入中拒绝特殊字符。因为有很多特殊字符在某些系统中拥有特殊含义，如“\”，在 Windows 系统中可能作为文件路径分隔符。在开发阶段这个问题可能不容易引起注意，但可能被攻击者利用。

### 3. 环境变量安全问题

环境变量的输入也可能给攻击者带来机会。主要来源于：

1) 环境变量的内容给攻击者留下机会。有些系统中，环境变量存储了和系统有关的一些配置信息，如在 Linux 中，很多程序配置被环境变量以某些隐含、模糊或未公开的方式所定义。例如，sh 和 bash shell 使用 IFS 变量来决定分隔命令行参数的字符。那么，在执行一个 shell 时，把 IFS 设置为某些值，就可能进行攻击。另一方面，由于并不是所有环境变量都有文档的说明，这让用户遇到攻击后无所适从；并且由于环境变量的可编辑性，攻击者甚至可以增加一些更危险的环境变量，从而达到攻击的目的。

2) 环境变量的存储格式给攻击者留下机会。在 Linux 下，环境变量在底层，通常是以字符串数组形式存储的，这个字符串指针数组有一个头指针，该数组中，按顺序存储各个环境变量，每个环境变量是这个数组中的一个元素，该数组以 NULL 指针结尾。每一个元素的格式都为: “NAME=value”。这潜在地表明，环境变量名不能包含等号，也不能包含一些其他敏感符号，如 NIL(ASCII 码为 0 的字符，一般表示字符串结尾)。如果这一点被攻击者利用，就可能会给系统带来损害。

要解决以上安全问题，可从以下几个方面着手：

- 限制环境变量的使用权限；
- 可适当破坏环境变量在 shell 之间的共享；
- 对用户定义的环境变量，需要进行严格的检查。

### 4. 文件名安全问题

在很多与文件输出有关的系统中，文件名可能成为安全隐患。无论在什么样的操作系统中，文件名应该遵循以下安全原则：

1) 不要让用户自己输入文件名，应在界面上给用户一个默认的文件名。如 Windows 中将文件另存时，界面的“文件名”框中会显示一个默认的合法名称，避免用户因为输入一些不合法的文件名造成安全问题。

2) 如果不得不让用户输入文件名，那么将文件名限制为只能包含字母和数字。特别应该考虑将一些特殊字符(如“/”、“\”、“-”，以及“*”、“？”、“[ ]”、“{ }”等通配符)从合法模式中去掉。

3) 不要允许用户命名一些可能和物理设备冲突的文件名。比如，在一些系统中，一些文件名可以被认为是物理设备。例如，如果一个程序试图打开 COM1 文件，可能被系统误解为尝试和串口通信，系统就去进行串口的读写，而该操作又不符合用户的期望。因此，这种文件命名也要避免。

### 5. 数据库安全问题

数据库(Database，DB)，顾名思义，是存放数据的地方。在计算机中，数据库包括两方面的含义：数据本身和数据库对象。不同的数据库产品，对数据都有不同的定义，但是展现给用户的数据库对象都类似，主要有表(Table)、视图(View)、存储过程(Stored Procedure)和触发器(Trigger)等。其中，表是最常用、最基本的数据库对象。关于这些数据库对象的定义，可以参考相关文档。

数据库一般用数据库管理系统(DBMS)进行管理，数据库管理系统是用于管理数据的计算机软件。利用数据库管理系统，用户能方便地定义和操纵数据，维护数据的安全性和完整性，以及进行多用户下的并发控制和恢复数据库，一般情况下，我们所说的“数据库软件”是指数据库管理系统。

目前，常用的数据库管理系统有 Access、Oracle、Sybase、FoxPro、DB2、Informix、SQL Server 等，它们在各种联机事务处理、数据仓库、电子商务、信息管理系统、决策支持系统、办公自动化系统、企业资源计划、网站建设等方面都有着广泛的应用。

攻击者通过对数据库的恶意输入，可将信息注入正在运行的进程，获取敏感数据，甚至危害进程的运行状态。

### 6. 账户和口令安全问题

在应用程序中，连接到数据库，通常要使用正确的账户和口令。但很多程序员直接用管理员账户连接到数据库，例如在 SQL Server 中，以 sa(Super Administrator，超级管理员账户)进行连接，这样很危险。在 Oracle 数据库中，以 system 进行连接也是很危险的，它们都是功能强大且很可能对各自系统造成损害的账户。

实际上，应用程序的使用，并不一定要用到管理员账户，使用管理员账户，反而给了攻

击者更多的机会。如在 SQL Server 中，当连接以 sa 账户进行，且 SQL 代码中有 bug 时，攻击者可执行任何管理员账户可以执行的任务，比如：删除系统中的数据库或表；删除系统中表中的数据；修改系统中表中的数据；修改存储过程、触发器；删除日志；添加新的数据库用户等。

很多情况下，程序员会将口令以明文形式存放于代码中，运行阶段，这些口令置入进程的内存空间。此时，口令如果被攻击者获知，则可执行攻击者希望执行的任何代码，危险性也很大。解决以上问题的方法主要有：尽量不使用管理员账户；使用最小特权账户，不给予额外的权限；不允许使用空口令连接数据库，防止管理员因疏忽而创建空口令；数据库连接字符串存放在配置文件中，最好可以加密，而不是在代码中以明文显示；发生错误时，仅给客户端通知信息，不给出具体原因，防止攻击者利用这些通知信息进行数据库猜测。

### 2.2.5　面向对象中的安全编程

面向对象(OO, Object Oriented)是目前最流行的软件开发方法之一，也是当前计算机软件工程界关注的重点，从 20 世纪 90 年代开始，它就慢慢成为软件开发方法的主流。目前，面向对象的概念和应用，已经不仅集中于程序设计和软件开发，而是扩充到计算机应用的其他应用场合，如数据库系统、交互式界面、应用结构、应用平台、分布式系统、网络管理结构、CAD 技术、人工智能等领域。面向对象强调人类在日常的思维逻辑中经常采用的思维方法与原则，其中的重要概念如抽象、分类、继承、聚合、多态等，都和我们的生活息息相关，这也成为面向对象思想流行的原因。对象的生成比较简单，涉及的安全考虑也不多；与此相对应，对象的内存也有释放过程，它与系统安全性的关系更大一些。

现以 C++为例，阐述对象在内存中的存储和释放情况。对象通常存放在三个内存区域：

1) 全局/静态数据区：主要存放全局对象和静态对象，在该内存区的对象或成员，直到进程结束，才会释放内存。

2) 堆：存在于堆中的数据，分配内存的方法一般是 new/malloc，释放内存的方法是 delete/free。对于这种对象，可对创建和销毁进行精确控制。堆对象在 C++中的使用非常广泛，也得到了广泛应用，不过，用这种方法分配或释放内存也有一些缺点：

- 需要程序员手工管理其创建和释放，如果忘记释放的话，可能造成内存泄漏；
- 在时间效率和空间效率上，堆对象没有栈对象高；
- 在程序中，如果频繁使用 new 来创建堆对象或用 delete 来释放堆对象，会造成大量的内存碎片，内存得不到充分使用。

3) 栈：栈中一般保存的是局部对象或局部变量，使用栈对象效率较高，程序员无需对其生存周期进行管理。

C++中，和对象释放内存相关的一般是析构函数。析构函数的作用是释放对象申请的资源，析构函数通常由系统自动调用，在以下几种情况下系统会调用析构函数：

- 全局对象在进程结束时；
- 堆中对象进行 delete/free 操作时；
- 栈中对象生命周期结束时：包括离开作用域、函数正常跳出或抛出异常等。

在使用析构函数时，可以充分利用它的性质进行一些操作，特别是对于栈中的对象，由于析构函数调用是由系统自动完成的，因此可以利用这一特性，将一些需要随着对象销毁而

必须释放的资源封装在析构函数中，由系统自动完成销毁或释放，这些工作的典型案例如：某些资源的释放、多线程解锁和关闭文件等。这样，利用栈对象的这一特性进行自动管理，可以避免由于编程时的遗漏而忘记执行某种操作。

很多情况下，对象可能在多线程环境下运行。一个对象在其生命周期内可以被多个线程访问，这实际上是多线程通信的一种方式，此种情况就会出现多种问题，其中最重要的就是多线程环境下对象的状态安全访问以及修改。实际上，很多系统软件(如服务器)已经在底层实现了线程安全，因此，隐患主要来源于：程序员不知道该组件对象使用线程来实现，错误地使用一些非线程机制情况下的方法。

关于对象线程安全，有两方面的问题：

1) 很多框架下都提供了对象被多个线程访问的机制，当对象可能被多个线程访问时，千万不能在对象中保存与某个线程相关的状态。

2) 当对象可能被多个线程进行操作时，应该考虑同步问题。

对象序列化是面向对象语言中的重要特性之一，在 Java 系列和.NET 系列中，都可以使用一定手段实现对象序列化。一般情况下，对象具有一定的生命周期，随着生成该对象的程序作用域结束而结束。但有时，程序可能需要将对象的状态保存下来，或者写入文件，或者写入数据传输流，在需要时再将对象读入之后进行恢复，这其中就需要序列化工作。对象序列化就是将对象的状态转换成字节流(当然也可能是字节流以上的一些包装流)，在使用时，可通过读取流中的内容，生成相同状态的对象。

序列化(Serialization)过程的工作一般是：对象将描述自己状态的数据写出到流中，描述自己状态的数据一般是成员变量，因此，序列化的主要任务是写出对象成员变量的数值。特殊情况下如果对象中某个成员变量是另一对象的引用，则被引用的对象也要序列化，因此，序列化工作是递归的。在很多应用中，对象序列化具有很重要的作用。如数据传输软件中，传输的数据一般是对象，这种情况下，对象应该具备写入流中的能力，也就是说需要被序列化；另外一些情况下，可能需要将对象写入持久化存储(如数据库和文件)，也需要进行对象的序列化。

以 Java 为例，将对象序列化的方法很简单，满足两个条件即可：

- 该对象对应的类实现 Serializable 接口；
- 该对象的被序列化成员必须是非静态成员变量；

其他语言中，序列化过程类似。不过，对象序列化只能保存成员变量的值，其他内容，如成员函数、修饰符等，都不能保存。

由于对象序列化之后要在网上传输，或者写入数据流，因此需要关心的问题一般是信息泄密问题。对象被序列化时，使用字节数组表示，并加上很多控制标记，在一定程度上阻止了对象成员直接被攻击者识别。但还是不能完全阻止对象内容的泄密，攻击者对保存的对象稍加分析，则可获取需要的信息。所以，在序列化时，一定要注意不能让敏感信息(如卡号密码)泄密。

解决该问题的方法有两种：

1) 在将对象实现序列化时，进行一些处理，如加密。该类方法将在后续章节中详细叙述。

2) 不要将敏感信息序列化。

可通过一些手段，不将某些成员序列化。如在 Java 中，可在成员前面加上 transient 关键

字，在序列化时，系统将回避这些字段。

在类中，数据成员可分为静态成员、非静态成员两种。

类中的成员，通常情况下，必须通过它的类的对象来访问，但可以创建这样一个成员，使它的使用完全独立于该类的任何对象，被类的所有对象共用。

在成员的声明前面加上关键字static(静态)就能创建这样的成员，这种成员叫做静态成员。如果一个成员变量或成员方法被声明为static，它就是静态变量或静态方法，它能够在该类的任何对象创建之前被访问，而不必引用任何对象。

静态成员变量存储在全局数据区，为该类的所有对象共享，不属于任何对象的存储空间，逻辑上所有对象都共享这一存储单元，对静态成员变量的任何操作都会影响这一存储单元的所有对象。

静态成员从属于一个类，不是某对象的一部分，非静态成员可直接访问类中的静态成员，但静态成员不可以访问非静态成员。

静态成员的优点是：消除传统程序设计方法中的全局变量，为真正实现封装性提供了必要手段。由于静态成员的共享性，就必须考虑其数据安全。

除了上一节讲到的线程安全外，还必须考虑对象对其进行访问时的安全性。要注意以下几点：

1) 静态成员的初始化操作先于对象的实例化而进行，所以在它们的初始化中不要启动线程，以免造成数据访问的问题，同时静态成员的初始化操作中也不应该有依赖关系；

2) 不用静态变量保存某个对象的状态，而应该保存所有对象共有的状态；

3) 不用对象来访问静态变量，而用类名来访问静态变量。

### 2.2.6　Web 编程安全

Web 编程是目前比较流行的软件编程方法之一，也是 B/S 模式的一种实现方式，由于 Web 编程的方法和传统 C/S 程序的不相同，因此 Web 编程中的安全问题也有其特殊性。

随着 Internet 技术的兴起，B/S 结构成为对 C/S 结构的一种改进。这种结构有如下特点：

- 程序完全放在应用服务器上，并在应用服务器上运行，通过应用服务器与数据库服务器进行通信；
- 客户机上不必安装任何客户端软件，系统界面通过客户端浏览器呈现，客户端只需在浏览器内输入 URL；
- 修改了应用系统，只需维护应用服务器。

由于 B/S 结构的优点，现在的网络应用系统中，B/S 系统占绝对主流地位。

了解了什么是 Web 程序，我们再深入了解一下 Web 技术的相关特点。在 Web 程序结构中，浏览器端与应用服务器端采用请求/响应模式进行交互。

过程描述如下：

1) 客户端(通常是浏览器，如 IE、Firefox 等)接受用户的输入，如用户名、密码、查询字符串等；

2) 客户端向应用服务器发送请求：输入之后提交，客户端把请求信息(包含表单中的输入以及其他请求等信息)发送到应用服务器端，客户端等待服务器端的响应；

3) 数据处理：应用服务器端使用某种脚本语言来访问数据库查询数据，并获得查询结果；

4) 数据库向应用服务器中的程序返回结果；

5) 发送响应：应用服务器端向客户端发送响应信息(一般是动态生成的 HTML 页面)；

6) 显示：由用户的浏览器解释 HTML 代码，呈现用户界面。

### 1. Web 编程

可以说，不同的 Web 编程语言都对应着不同的 Web 编程方式，目前常见的应用于 Web 的编程语言可以参考 5.4.1 一节的相关内容。

### 2. 避免 URL 攻击

Web 上有很多资源，如 HTML 文档、图像、视频、程序等，在访问时，它们的具体位置怎样确定呢？通常是利用 URL。

URL(Uniform Resource Locator，统一资源定位器)是 Internet 上用来描述信息资源的字符串，可帮助计算机来定位这些 Web 上的可用资源。

以下是一个典型的 URL 例子：http://localhost:8080/Prj08/index.jsp?username=gkhua。可以看出，URL 一般由 4 部分组成：

- 访问资源的命名机制(协议)：http，实际上还可能是 ftp 等；
- 存放资源的主机名：localhost:8080；
- 资源自身的名称，由路径表示：/Prj08/index.jsp；
- 其他信息，如查询字符串等：?username=gkhua。

URL 操作攻击的原理，一般是通过 URL 来猜测某些资源的存放地址，从而非法访问受保护的资源。

为防止 URL 操作攻击，程序员在编写 Web 应用时，可从以下方面加以注意：

- 为避免非登录用户进行访问，对于每一个只有登录成功才能访问的页面，应该进行 session 检查(session 检查的内容将在稍后提到)；
- 为限制用户访问未被授权的资源，可在查询时将登录用户的用户名也考虑进去。

### 3. 页面状态值安全

HTTP 是无状态的协议，除非通过服务器，Web 页面本身无法向下一个页面传递信息。因此，Web 页面保持状态并传递给其他页面是一个重要技术。

举一个简单的案例：页面 1 中定义了一个数值变量，并通过计算，将其平方值显示在界面上；同时页面 1 中有一个链接到页面 2，要求点击链接，在页面 2 中显示该数字的立方。很明显，该应用中，页面 2 必须知道页面 1 中定义的那个变量。

在 HTTP 协议中共有 4 种方法来完成这件事情：

- URL 传值
- 表单传值
- Cookie 方法
- session 方法

这 4 种方法各有特点，各有安全优势，本节将对其进行分析。

URL 方法由于其简单性和平台支持的多样性(没有浏览器不支持 URL)，很多程序使用 URL 传值比较方便。

但该方法有如下问题：

- 传输的数据只能是字符串，对数据类型具有一定限制；
- 传输数据的值会在浏览器地址栏里被看到。从保密角度看，这是不安全的。特别是安全性要求很严格的数据(如密码)不应该用 URL 方法来传值。

与 URL 一样，表单传值方法简单方便，支持多平台，很多程序使用这种方法。

但该方法有如下问题：

- 与 URL 方法类似，该方法传输的数据也只能是字符串，对数据类型具有一定限制；
- 传输数据的值虽然可以保证在浏览器地址栏内不被看到，但在客户端源代码里会被看到。因此，从保密角度看，这也是不安全的。特别是秘密性要求很严格的数据(如密码)，也不推荐用表单方法来传值。

在页面之间传递数据的过程中，Cookie 是一种常见的方法。Cookie 是一个小的文本数据，由服务器端生成，发送给客户端浏览器，客户端浏览器如果设置为启用 Cookie，则会将这个小文本数据保存到其某个目录下的文本文件内。

客户端下次登录同一网站，浏览器则会自动将 Cookie 读入之后，传给服务器端。服务器端可对该Cookie进行读取和验证(当然也可以不读取)。一般情况下，Cookie中的值以 key-value 的形式进行表达。

基于这个原理，上面的例子可以用 Cookie 来进行。即在第一个页面中，将要共享的变量值保存在客户端 Cookie 文件内，在客户端访问第二个页面时，由于浏览器自动将 Cookie 读入之后，传给服务器端，因此只需要第二个页面中由服务器端页面读取这个 Cookie 值即可。在客户端的浏览器上，我们看不到任何与传递的值相关的信息，说明在客户端浏览器中，Cookie 中的数据是安全的。

但也不能说 Cookie 是完全安全的。因为 Cookie 是以文件形式保存在客户端的，客户端存储的 Cookie 文件就可能被攻击者获知。在本例中，内容被保存在 Cookie 文件中，如果使用的是 Windows XP，C 盘是系统盘，该文件保存在 C:\Documents and Settings\当前用户名\Cookies 下。很明显，Cookie 也不是绝对安全的。如果将用户名、密码等敏感信息保存在 Cookie 内，在用户离开客户机时不注意清空，这些信息容易泄露，因此 Cookie 在保存敏感信息方面具有潜在危险。

可以很清楚地看到，Cookie 的危险性来源于 Cookie 可能被盗取。目前盗取的方法有很多种：

1) 利用跨站脚本技术，将信息发给目标服务器；为了隐藏跨站脚本的 URL，甚至可以结合 Ajax(异步 JavaScript 和 XML 技术)在后台窃取 Cookie。

2) 通过某些软件，窃取硬盘下的 Cookie。如前所述，当用户访问完某站点后，Cookie 文件会保存在机器的某个文件夹(如 C:\Documents and Settings\用户名\Cookies)下，因此可以通过某些盗取和分析软件来盗取 Cookie。具体步骤如下：

- 利用盗取软件分析系统中的 Cookie，列出用户访问过的网站；
- 在这些网站中寻找攻击者感兴趣的网站；
- 从该网站的 Cookie 中获取相应信息。

不同的软件有不同的实现方法，有兴趣的读者可在网上搜索相应的软件。

3) 利用客户端脚本盗取 Cookie。在 JavaScript 中有很多 API 可以读取客户端 Cookie，可

将这些代码隐藏在一个程序(如图片)中，很隐秘地得到 Cookie 值，不过这也是跨站脚本的一种实现方式。

同样，以上问题不代表 Cookie 就没有任何用处，Cookie 在 Web 编程中还是应用很广泛的，主要是因为以下几个方面：

1) Cookie 的值能够持久化，即使客户端机器关闭，下次打开还是可以得到里面的值。因此 Cookie 可用来减轻用户一些验证工作的输入负担，比如用户名和密码的输入，就可以在第一次登录成功之后，将用户名和密码保存在客户端 Cookie，下次不必输入。当然，这不安全，但对于一些安全要求不高的网站，Cookie 还是大有用武之地。

2) Cookie 可以帮助服务器端保存多个状态信息，但不用服务器端专门分配存储资源，减轻了服务器端的负担。比如网上商店中的购物车，必须将物品和具体客户名绑定，但是放在服务器端又需要占据大量资源的情况下，可以用 Cookie 来实现，将每个物品和客户的内容作为 Cookie 保存在客户端。

3) Cookie 可持久保持一些与客户相关的信息。如很多网站上，客户可自主设计自己的个性化主页，其作用是避免用户每次都需要去找自己喜爱的内容，设计好后，下次打开该网址，主页上显示的是客户设置好的界面。如果这些设置信息保存在服务器端，就会消耗服务器端资源，因此可将客户的个性化设计保存在 Cookie 内，每一次访问该主页，客户端将 Cookie 发送给服务器端，服务器根据 Cookie 的值来决定显示给客户端什么样的界面。

解决 Cookie 安全的方法有很多，常见的有以下几种：

1) 替代 Cookie。将数据保存在服务器端，可选的是 session 方案；

2) 及时删除 Cookie。要删除一个已经存在的 Cookie，有以下几种方法：

- 给一个 Cookie 赋空值；
- 设置 Cookie 的失效时间为当前时间，让该 Cookie 在当前页面浏览完之后就被删除了；
- 通过浏览器删除 Cookie。如在 IE 中，可以选择“工具” | “Internet 选项” | “常规”，在其中单击“删除 Cookies”，就可以删除文件夹中的 Cookie。

3) 禁用 Cookie。很多浏览器中都设置了禁用 Cookie 的方法，如 IE 中，可在“工具” | “Internet 选项” | “隐私”中，将隐私级别设置为禁用 Cookie。

前面几种方法在传递数据时，有一个共同的问题是内容都保存在客户端，具有泄露的危险。如果在不考虑服务器负载的情况下，将数据保存在服务器端是一个较好的方案，这就是 session 方法。

本质上讲，会话(session)的含义是指某个用户在网站上的有始有终的一系列动作的集合。例如，用户在访问网站时，session 就是指从用户登入站点到关闭浏览器所经过的这段过程。session 中的数据可被同一个客户在网站的一次会话过程共享。但是对于不同客户来说，每个人的 session 是不同的。可见，也可以实现页面之间数据的传递。session 方法和前面几个方法相比，是相对安全的。

同一个客户在访问多个页面时，多个页面用到 session，对他来说是同一个对象。那么服务器怎么知道要分配给它的是同一个 session 对象呢？实际上，在客户进行第一次访问时，服务器端就给 session 分配了一个 sessionId，并让客户端记住了这个 sessionId，客户端访问下个页面时，又将 sessionId 传送给服务器端，服务器端根据这个 sessionId 找到前一个页面用的 session，由此保证为同一个客户服务的 session 对象是同一个。

综上所述，session 分配的具体过程为：

- 客户端访问服务器，服务器使用 session，首先检查这个客户端的请求中是否已包含 sessionId；
- 如果有，服务器就在内存中检索相应 Id 的 session 来用；
- 否则服务器为该客户端创建一个 session 并生成一个相应的 sessionId，并在该次响应中返回给客户端保存。

session 经常用于保存用户登录状态。比如用户登录成功之后要访问好几个页面，但是每个页面都需要知道是哪个用户在登录，此时就可以将用户的用户名保存在 session 内。

在大项目中，有许多页面可能都要进行 session 检查，如果将 session 检查代码写很多次，势必出现大量重复代码。针对该问题，可用两种方法加以解决(此处只是列举 JSP 系列中的解决办法)：

1) 将 session 检查代码单独写入一个文件，在每个需要检查的网页中包含它。

2) 利用过滤器。不过，并不是所有的语言都支持过滤器。关于过滤器的知识，读者可以查阅相关文献。

以上内容主要是站在编程角度来谈论 session 应该怎样使用才最安全；那么，针对 session 的攻击主要体现在哪里呢？

session 机制最大的不安全因素是 sessionId 可被攻击者截获，如果攻击者通过一些手段知道了 sessionId，由于 sessionId 是客户端寻找服务器端 session 对象的唯一标识，攻击者就有可能根据 sesionId 来访问服务器端的 session 对象，得知 session 中的内容，从而实施攻击。

在 session 机制中，很多人认为：只要浏览器关闭，会话结束，session 就消失了。其实不然，浏览器关闭，会话结束，对于客户端来说，已经无法直接再访问原来的那个 session，但并不代表 session 在服务器端会马上消失。除非程序通知服务器删除一个 session，否则服务器会一直保留这个 session 对象，直到 session 超时失效，被垃圾收集机制收集掉。

令人遗憾的是，客户在关闭浏览器时，一般不会通知服务器。由于关闭浏览器不会导致 session 被删除，因此客户端关闭后，session 还未失效的情况下，就给攻击者留下获取 session 中的内容的机会。

虽然 sessionId 是随机的长字符串，通常比较难被猜测到，这在某种程度上可以加强其安全性，但一旦被攻击者获得，就可以进行一些攻击活动，如攻击者获取客户 sessionId，然后攻击者自行伪造一个相同的 sessionId，访问服务器，实际上等价于伪装成该用户进行操作。

为防止以上因为 sessionId 泄露而造成的安全问题，可以采用如下方法：

1) 在服务器端，可在客户端登录系统时，尽量不要使用单一的 sessionId 对用户登录进行验证。可以通过一定的手段，不时地变更用户的 sessionId；

2) 在客户端，应该在浏览器关闭时删除服务器端的 session，也就是说在关闭时必须通知服务器端。最简单的方法是用 JavaScript 实现。

#### 4. Web 跨站脚本攻击

跨站脚本在英文中称为 Cross-Site Scripting，缩写为 CSS。但由于层叠样式表(Cascading Style Sheets)的缩写也为 CSS。因此，为了将两者相区别，特将跨站脚本缩写为 XSS。

跨站脚本，顾名思义，就是恶意攻击者利用网站漏洞往 Web 页面里插入恶意代码，一般需要以下几个条件：

- 客户端访问的网站是一个有漏洞的网站，但没有意识到；
- 在这个网站中通过一些手段植入一段可以执行的代码，吸引客户执行(通过鼠标点击等)；
- 客户点击后，代码执行，达到攻击目的。

XSS 属于被动式攻击。XSS 可诱使 Web 站点执行本来不属于它的代码，而这些代码由攻击者提供，攻击者利用这些代码执行来获取信息。XSS 涉及三方，即攻击者、客户端与客户端访问的网站。XSS 的攻击目标是盗取客户端的敏感信息。从本质上讲，造成 XSS 漏洞的原因是网站的 Web 应用对用户提交请求参数未做充分的检查过滤。XSS 攻击的主要危害包括：盗取用户的各类敏感信息，如账号密码等；读取、篡改、添加、删除企业敏感数据；读取企业重要的具有商业价值的资料；控制受害者机器向其他网站发起攻击等。一些比较著名的网站，如 eBay，也曾遭受过 XSS 攻击，有兴趣的读者可参考相关资料。

如何防范 XSS 攻击呢？主要从网站开发者角度和用户角度来阐述。

1) 从网站开发者角度，需要：

根据来自 OWASP(开放应用安全计划组织)的建议，对 XSS 最佳的防护主要体现在以下两个方面：

- 对于任意的输入数据应该进行验证，以有效检测攻击；也就是说，某个数据被接受之前，必须使用一定的验证机制来验证所有输入数据，如长度、格式、类型、语法等；常见的方法，比如黑名单验证，就是将一些常见的字符(如“<>”或类似“script”的关键字)进行过滤，效果比较好；不过，该方式也有局限性，很容易被 XSS 变种攻击绕过验证机制。
- 对于任意的输出数据，要进行适当的编码，防止任何已成功注入的脚本在浏览器端运行；数据输出前，确保用户提交的数据已被正确进行编码；可在代码中明确指定输出的编码方式(如 ISO-8859-1)，而不是让攻击者发送由他自己编码的脚本给用户。

2) 从网站用户角度，需要：

- 打开一些 Email 或附件、浏览论坛帖子时一定要特别谨慎，否则有可能导致恶意脚本执行。可在浏览器设置中关闭 JavaScript 来防止执行恶意脚本，如下所示，如果是 IE 的话，可单击“工具”|“Internet 选项”|“安全”|“自定义级别”进行设置。
- 增强安全意识，只信任值得信任的站点或内容，不要信任别的网站发到自己信任的网站中的内容。
- 使用浏览器中的一些配置。

### 5. SQL 注入攻击的危害

SQL 注入在英文中称为 SQL Injection，是黑客对 Web 数据库进行攻击的常用手段之一。在这种攻击方式中，恶意代码被插入到查询字符串中，然后将该字符串传递到数据库服务器执行，根据数据库返回的结果，获得某些数据并发起进一步攻击，甚至获取管理员账号密码、窃取或篡改系统数据。

SQL 注入攻击的主要危害包括：

- 非法读取、篡改、添加、删除数据库中的数据；

- 盗取用户的各类敏感信息，获取利益；
- 通过修改数据库来修改网页上的内容；
- 私自添加或删除账号；
- 注入木马。

由于 SQL 注入攻击一般利用的是 SQL 语法，这使得所有基于 SQL 语言标准的数据库软件(如 SQL Server、Oracle、MySQL、DB2 等)都可能受到攻击，并且攻击的发生和 Web 编程语言本身也无关，如 ASP、JSP、PHP，在理论上都无法完全幸免。

SQL 注入攻击的危险是比较大的。其他很多攻击，如 DoS 等，可能通过防火墙等手段进行阻拦，但是对于 SQL 注入攻击，由于注入访问是通过正常用户端进行的，因此普通防火墙对此不会发出警示，一般只能通过程序来控制，而 SQL 攻击一般可以直接访问数据库，进而甚至获得数据库所在的服务器的访问权，因此危害相当严重。

以上问题的解决方法有很多种，比较常见的有：

1) 将输入中的单引号变成双引号。这种方法常用于解决数据库输入问题，同时也是一种对于数据库安全问题的补救措施。

2) 使用存储过程。

3) 认真对表单输入进行校验，从查询变量中滤去尽可能多的可疑字符。

可以利用一些手段，测试输入字符串变量的内容，定义一种格式为只接受的格式，只有此种格式下的数据才能被接受，拒绝其他输入的内容，如二进制数据、转义序列和注释字符等。另外，还可对用户输入的字符串变量进行类型、长度、格式和范围验证并过滤，也有助于防止 SQL 注入攻击。

4) 在程序中组织 SQL 语句时，应该尽量将用户输入的字符串以参数形式进行包装，而不是直接嵌入 SQL 语言。

由于很多 SQL 注入都把用户输入和原始的 SQL 语言嵌套组合成查询语句来完成攻击，而参数不能被嵌套进入 SQL 查询语言，因此该种措施可以在一定程度上防止 SQL 注入。不过，在不同的语言和产品里面，做法稍有不同，如果使用 Java 系列，可使用 PreparedStatement 代替 Statement；SQL Server 数据库中可以使用存储过程，结合 Parameters 集合；Parameters 集合提供了长度验证和类型检查的功能，Parameters 集合内的内容将被视为字符值而不是可执行代码；.NET 中，SQL 语句可以用参数来包装等。

5) 严格区分数据库访问权限。

在权限设计中，应用软件的使用者一定要限制权限，没必要赋予他们对数据库对象的建立、删除等权限。这样即使在收到 SQL 注入攻击时，有一些对数据库危害较大的操作，如 DROP TABLE 等语句，也不会执行，可最大限度地减少注入式攻击对数据库带来的危害。

6) 多层架构下的防治策略。

在多层环境下，用户输入数据的校验与数据库的查询被分离成多个层次。此时应该采用以下方式进行验证：

- 用户输入的所有数据，都需要进行验证，通过验证才能进入下一层，此过程与数据库分离。
- 没有通过验证的数据应该被数据库拒绝，并向上一层报告错误信息。

7) 对于数据库敏感的、重要的数据，不要以明文显示，要进行加密。关于加密的方法，

读者可以参考后续章节。

8) 对数据库查询中的出错信息进行屏蔽，尽量减少攻击者根据数据库的查询出错信息来猜测数据库特征的可能。

9) 由于 SQL 注入有时伴随着猜测，因此如果发现一个 IP 不断进行登录或短时间内不断进行查询，可以自动拒绝此 IP 的登录；也可以建立攻击者 IP 地址备案机制，对曾经的攻击者 IP 进行备案，发现此 IP，直接拒绝。

10) 可以使用专业的漏洞扫描工具来寻找可受到被攻击的漏洞。

#### 6. 避免 Web 认证攻击

在Web 应用程序的安全中，认证是一个重要的角色。对于一些受保护的资源，必须在身份验证的基础上进行。认证实际上属于权限控制的一种，前面的章节阐述了权限控制方面的一些内容，与传统的认证方式类似，Web 上流行的认证类型主要有以下几种，可以归纳为以下 3 类：

1) 用户名/密码。因为简单，该种认证方法实际上是当今 Web 上最流行的认证形式。一般情况下，可提供表单让用户输入用户名/密码；

2) 其他认证方法。用户名/密码方法也有一些脆弱性，于是发明了一些更强健的认证方法，如基于令牌和基于证书的认证，许多 Web 站点都开始为客户提供这种认证方法；

3) 认证服务。许多大公司提供了专门的认证服务，如微软的 Passport，实现了一个私有信息的管理和认证协议；而一些 Web 站点把其上用户的认证外包给这些认证服务公司。

Web 认证攻击，主要过程是利用传统 Web 认证机制的漏洞，以各种攻击形式获取合法用户的身份信息(如用户名/密码)，从而访问某些资源、盗取用户信息或控制用户程序；还有一种形式，就是利用漏洞来绕过 Web 认证，对合法用户的信息进行修改，或者进行恶意的数据传输。

Web 认证攻击的方法有很多，用户可根据实际情况采用不同的手段加以应对。常见的 Web 认证攻击方法有：

- 用户名枚举
- 密码猜测
- 密码窃听和重放攻击等

实际上，Web 认证攻击在很多情况下可通过 SQL 注入、窃听等方式来实行，对于不同的 Web 站点，无法说明哪种方法最好，因此此处提出几个安全准则：

1) 密码在数据库中不以明文存储，可以进行加密，如用 MD5 算法进行加密等，可让攻击者无法直接得知密码；

2) 建立较好的密码策略和账户锁定策略，如使用者如果多次尝试某个密码，可以锁定其账户；

3) 在账号和密码的传输中，使用 HTTPS 方法来保护认证的传输，这样可以较大程度上避免受到窃听和重放攻击的风险；

4) 进行严格的输入验证。这是一个常见的主题，可有效防范很多种类的攻击，如跨站脚本、SQL 注入、命令执行等。

### 2.2.7　远程过程调用安全

远程过程调用(RPC，Remote Procedure Call)给程序功能的扩充提供了较大的支持。远程过程调用为程序的分布式应用开发架构提供了技术支持，不需要了解底层网络通信协议，在应用层通过网络从远程计算机程序请求服务。

传统的网络分布式程序需要进行复杂的底层通信编程，但有了远程过程调用后，开发网络分布式应用程序更容易了。RPC 的出现，让开发者不需要了解底层网络通信协议，直接通过网络从远程计算机程序请求服务。

RPC 通信模型基于客户端/服务器通信模型，是一种同步通信方式，即调用方必须等待服务器响应。在客户端，RPC 为远程过程提供了抽象，在调用时，其底层消息传递机制对客户端来说都是透明的。

在 RPC 中，服务以过程形式放在服务器端，客户端负责请求服务，服务器执行客户端的请求，运行被调用的过程。

RPC 在整个调用过程中需要经历的步骤如下：

- 客户端请求进行远程调用，激活客户端存根，指定目标服务器；
- 客户端存根将被调用的过程和参数打包，作为消息发送给服务器，等待数据消息的返回；
- 服务器接收消息，服务器存根根据消息中的过程和参数等信息，调用服务器端的过程；
- 服务器将结果作为消息返回给客户端存根；客户端存根将结果返回给用户。

客户端使用桌面应用程序，很显然，为应对数据库的迁移或改变，访问数据库的代码不应写在客户端，否则会造成大量客户端的改变。此时，访问数据库的代码写在服务器端，作为方法或过程的形式对外发布，客户端可在不知道服务器细节，也不知道底层通信协议的基础上，访问服务器端的这些方法，就像调用自己机器上的方法一样。客户端不必知道服务器端的核心代码，只需知道接口即可。

RPC 的好处：

1) 给程序在异构环境下进行通信提供了可能，异构主要可以体现在：

- 网络环境中的多种硬件系统平台；
- 硬件平台上的不同系统软件；
- 不同的网络协议或网络体系结构连接等。

2) 在异构网络环境下，需要把分散在各地的计算机系统集成起来，充分利用系统中分散的计算资源，由网络中的多台计算机协同工作完成某一任务，RPC 也给这种需求的实现提供了可能。

RPC 具有强大的网络编程功能，给编程带来了极大方便，并为分布式计算提供了支持，但仍存在一些安全问题。主要体现在：

1) 攻击者可能恶意调用 RPC 服务器中的过程，或输入一些恶意的数据导致服务器失效。

由于 RPC 处理过程中，底层使用的仍是 TCP/IP 协议，而 TCP/IP 协议本身存在缓冲区溢出的问题，攻击者就可能利用这一漏洞，对系统进行攻击。一般情况下，RPC 使用的是 135 端口(RMI 使用的是 1099 端口)。攻击者可伪装成合法客户端，向 RPC 端口传送信息，并让该信息溢出服务器端的 RPC 缓冲区，如果客户端发送的信息经过了精心设计，那么足可以

执行恶意的代码。

通常，如果服务器被攻击，一些基于 RPC 的服务，如 DCOM(Distributed Component Object Model，分布式组件对象模型)都将无法正常运行。更有甚者，攻击者有可能获得对远程服务器的完全控制，对服务器随意进行操作，如安装程序、篡改数据、格式化硬盘、创建用户或增加权限等。

通常利用以下方法来解决此问题：

- 利用防火墙封堵端口。可以设置防火墙的分组过滤规则，过滤掉 RPC 端口和影响到 DCOM 函数调用的数据包，通过这种方法，可以避免防火墙内的系统被外部攻击。
- 临时禁用某些服务，如 DCOM。如果因为一些特殊原因无法过滤 RPC 端口，也可临时关闭 DCOM 服务来保证网络安全。不过，该方法将导致系统运行异常，因此一般不建议使用。有关方法可参考相关文档。

2) 客户端和服务器之间传递的信息可能被窃听；攻击者可能对传输中的数据进行篡改。

在 RPC 通信机制中，调用组件和返回客户信息都通过传送消息进行，由于消息在传送过程中采取的安全措施比较简单，很容易被非法用户截获，造成信息泄密。

为保证网络系统中的消息信息的安全，可采用数据加密和解密的方法来实现。这里可采用加密解密与数字签名来实现，该方法在后续章节中将有详细叙述。

## 2.3　本章小结

本章首先讲述 SSDL 的相关知识。为将测试计划的利益最大化，SSDL 与系统开发生命周期进行了结合。在项目起始阶段，相关的 SSDL 阶段称作“安全原则、规则及规章”。这些是必须在早期进行定义，而且被当作安全性实现的保护性原则。

安全需求在业务分析和需求阶段被定义。安全需求的定义涉及所有的风险承担者、需要交流的术语以及需要进行教育的人。

在原型、设计和架构阶段，安全小组通过指导架构评审、威胁建模来指出所有潜在的安全漏洞，从而为这一阶段的工作提供支持。

安全编码原则有助于防止在代码中潜入安全漏洞。在软件开发中，应始终坚持这些原则。白盒、黑盒和灰盒测试技术将结合集成和测试阶段来开展工作。在首次软件构建基本形成之后，将开展系统测试。

判定可利用性在整个生命周期都会进行，该活动将在系统开发为产品并进入维护阶段期间最终定案。在整个测试生命周期内，都需要对安全项目评审和评估活动进行指导，从而开展持续改进活动。

安全部署所需考虑的事项必须实现。随着安全测试的展开，可对度量标准进行评价，同时还需要对最终评审和评估活动给予指导，从而能够做出有充分依据的决策。

在生命周期的最后，应完成安全的应用程序部署。安全的部署意味着软件使用了安全的默认值进行安装，文件的许可权限进行了恰当的设置，并在应用程序的配置中使用了安全的设置。

必须在维护阶段做好判定可利用性、判定漏洞及补丁管理工作。如果在开发时觉得某部分

会成为安全问题，就应该立即对代码进行修正，从而消除所有可能成为安全问题的疑虑。但是，如果软件已经部署，那么进行一次修正的代价就变得非常高昂。这种情况下，应使用判定可利用性技术，免得在绝对必要的情况之外，给用户带来额外的花费和工作。

本章随后主要对软件编程的安全方面进行了介绍，包括内存安全、进程与线程安全、异常与错误处理、输入安全、面向对象中的安全、Web 编程安全和远程调用安全的知识。

内存安全主要讲解缓冲区溢出问题，对攻击方式和应对方法进行了介绍。讨论了进程与线程安全的概念与区别，以及死锁的概念。异常与错误是编程中经常要考虑的问题，对异常与错误的处理至关重要。对于输入问题，无论是外部输入，还是程序内部数据的传递，都要进行安全性检验。面向对象安全中阐述了对象的内存释放过程的安全和对象序列化的问题。Web 编程安全介绍了 Web 编程中的 URL 攻击、页面状态值、跨站脚本攻击、SQL 注入攻击以及 Web 认证攻击的基本知识。最后讲述了远程过程调用的攻击方式和应对方法。

# 第3章　软件安全技术

## 本章导读

本章将介绍软件安全技术的知识。加密技术部分介绍常见加密方法、原理和密钥管理。认证技术部分介绍认证及身份验证技术和访问控制技术。最后讲解安全保障和网络安全方面的知识。

应掌握的知识要点：

- 加密技术；
- 加密方法与技术；
- 密钥管理方法与技术；
- 身份验证技术；
- 访问控制技术；
- 安全保障；
- 防火墙技术；
- 入侵检测；
- 安全扫描。

## 3.1　加密技术

在信息时代，数据的安全越来越受到关注。对于保存在计算机上的某些数据，我们希望其信息不被人所知；对于在网络上传输的重要数据，我们希望即使被敌方窃听也不会泄密。此时，将信息进行加密，就成了保障数据安全的首要方法。

加密算法一般可分为对称加密、非对称加密和单向加密三类，由于特点不同，在不同系统中具有不同的应用范围，各类算法中都有一些代表性算法。如对称加密体系中的DES、3DES、AES 算法；非对称加密体系中的 RSA、DSA 算法；单向加密中的 MD5、SHA 算法等。

通常加密体系中最核心的内容是加密算法和密钥；加密算法通常公开，加密系统的安全性取决于密钥的隐蔽性。因此，密钥管理是加密系统中的重要工作。

### 3.1.1　加密概述

我们知道，信息被敌方获取并得知，可能被敌方用于非法的操作以获取利益；加密是以某种特殊算法改变原有的信息数据，这种情况下，未授权的用户即使获得了已加密的信息，但是因为无法知道解密的方法，仍然无法了解信息的内容。目前，数据加密技术已经广泛应

用于 Internet 电子商务、手机网络和银行自动取款机等领域。加密系统中有如下重要概念：

- 明文(plaintext)：需要被保护的消息；
- 密文(ciphertext)：将明文利用一定算法进行改变后的消息；
- 加密(encryption)：将明文利用一定算法转换成密文的过程；
- 解密(decryption)：由密文恢复出明文的过程；
- 敌方：也称攻击者，通过各种办法，窃取机密信息来达到非法目的；
- 被动攻击(passive attack)：获取密文，目的是经过分析得到明文，这是一类攻击的总称；
- 主动攻击(active attack)：非法入侵者采用篡改、伪造等手段向系统注入假消息等，这也是一类攻击的总称；
- 加密算法：对明文进行加密时采用的算法；
- 解密算法：对密文进行解密时采用的算法。

这里特别需要强调的一个概念是“密钥(key)”。密钥包括加密密钥(encryption key)和解密密钥(decryption key)。由于加密算法和解密算法的操作通常是在一组输入数据的控制下进行的，这组输入数据就叫做密钥，在加密时使用的密钥为加密密钥，解密时使用的密钥为解密密钥。

在密码系统中，有两大要素：密码算法(加密算法和解密算法)和密钥。两者之间具有紧密的联系。以最简单的“恺撒加密法”为例：《高卢战记》中描述，恺撒大帝曾使用密码来传递信息，即所谓的“恺撒密码”。它是一种替代密码，通过将字母按顺序推后 3 位起到加密作用。如将字母 A 换作字母 D，将字母 B 换作字母 E，X、Y、Z 字母分别又变为 A、B、C 字母。如“China”可以变为“Fklqd”；解密过程相反。在这个简单的加密方法中，“向右移位”可以理解为加密算法；“3”可以理解为加密密钥。对于解密过程，“向左移位”，可以理解为解密算法；“3”可以理解为解密密钥。显然，密钥是一种参数，它是在明文转换为密文或将密文转换为明文的算法中输入的数据。恺撒加密法的安全性来源于两个方面：对密码算法本身的保密；对密钥的保密。

只针对密码算法进行保密，以保护信息，在学界和业界已有相当多的讨论，一般认为是不够安全的。目前业界中广泛认为，加密之所以安全，是因为其密钥的保密，并非加密算法本身的保密。因此，密码算法一般公开，而将密钥进行保密。如果攻击者要通过密文得到明文，除非对每一个可能的密钥进行穷举性测试。从后面的篇幅将可以看出，流行的一些加密解密算法一般是完全公开的。敌方如果取得已加密的数据，即使得知加密算法，若没有密钥，也无法进行解密。

### 3.1.2　加密方法及技术

加密技术从本质上说是对信息进行编码和解码的技术。加密将可读信息(明文)变为代码形式(密文)；解密是加密的逆过程，相当于将密文变为明文。加密算法有很多种，这些算法一般可分为三类：对称加密、非对称加密和单向加密。

#### 1. 对称加密

在对称加密算法中，双方使用的密钥相同，要求解密方事先必须知道对方使用的加密密钥。其算法一般公开，优势是计算量较小、加密速度较快、效率较高。不足之处是，通信双方使用同样的密钥，密钥在传送的过程中，可能被敌方获取，安全性得不到保证。当然，为

安全起见，用户每次使用该算法，可以更换密钥，但是原来通信的密钥也不能马上删除，但是这样使得双方所拥有的密钥数量很大，对于双方来说，密钥管理较为困难。

对称加密算法应用较早，技术较成熟。过程如下：发送方对明文用加密密钥和加密算法进行加密处理，变成密文，连同密钥一起，发送给接收方；接收方收到密文后，使用发送方的加密密钥及相同算法的逆算法对密文解密，恢复为明文。

DES 是数据加密标准(Data Encryption Standard)的简称，来源于 IBM 的研究工作，并在 1977 年被美国政府正式采纳。这种密钥系统使用得非常广泛，最初开发 DES 是嵌入硬件中，后来在其他领域得到了发展。在金融数据安全保护等领域，DES 发挥了巨大作用。

DES 算法的基本思想如下：首先确定一个 64 位的初始密钥 K(也称为主密钥)，在这 64 位中，实际密钥只有 56 位，另有 8 位是奇偶校验位，分布于 64 位密钥中，每 8 位中有 1 位奇偶检验位。要对 DES 进行攻击，一般只能使用穷举密钥搜索方法，即重复尝试各种密钥，直到找到符合的为止，这种做法无法实现。

3DES，即三重 DES，是 DES 的加强版，也是 DES 的一种更安全的变形。在原理上，它使用 3 个 56 位(共 168 位)的密钥对数据进行三次加密，和 DES 相比，安全性得到了较大的提高。实际上，3DES 是一个过渡的加密算法。1999 年，NIST 将 3DES 指定为 DES 向 AES 过渡的加密标准。3DES 以 DES 为基本模块，通过组合分组方法设计出分组加密算法。若三个密钥互不相同，本质上就相当于用一个长为 168 位的密钥进行加密，大大加强了数据的安全性。若数据对安全性要求不高，可让其中的两个密钥相等，这样，密钥的有效长度也有 112 位。

AES 在密码学中是高级加密标准(Advanced Encryption Standard)的缩写，该标准是 NIST 推出旨在取代 DES 的 21 世纪的加密标准，已被多方分析且广为使用，在对称加密系统中成为最流行的算法之一。AES 算法已被广泛应用于各个领域。AES 设计的密钥长度有 128 位 192 位、256 位三种，比 DES 的 56 密钥安全性更强。

### 2. 非对称加密

与对称加密算法不同，非对称加密算法需要两个密钥：公开密钥(public key)和私有密钥(private key)。每个人都可以产生这两个密钥，其中，公开密钥对外公开(可以通过网上发布，也可以传输给通信的对方)，私有密钥不公开。对于同一段数据，利用非对称加密算法具有如下性质：如果用公开密钥对数据进行加密，那么只有用对应的私有密钥才能对其解密；如果用私有密钥对数据对其进行加密，那么只有用对应的公开密钥才能对其解密。

非对称加密算法的基本过程是：首先，通信前，接收方随机生成一对公开密钥和私有密钥，将公开密钥公开给发送方，自己保留私有密钥；然后发送方利用接收方的公开密钥加密明文，使其变为密文；最后接收方收到密文后，使用自己的私有密钥解密密文，获得明文。

目前，在非对称密码体系中，使用得比较广泛的非对称加密算法有 RSA 和美国国家标准局提出的 DSA 等。与对称加密算法相比，非对称加密算法的保密性比较好，在通信过程中，只有公开密钥在网络上传输，即使公开密钥被敌方获取，也没有用；因此，基本不必担心密钥在网上被截获而引起安全问题。但该加密体系中，加密和解密花费时间较长、速度较慢，一般情况下，不适合对大量数据进行加密，只适用对少量数据进行加密。

RSA 算法出现于 20 世纪 70 年代，既能用于数据加密，也能用于数字签名。由于其易于理解和容易操作而广泛流行。该算法由 Ron Rivest、Adi Shamir 和 Leonard Adleman 发明，也

就以三人的名字命名。针对 RSA 的研究比较广泛，在使用过程中，经历了各种攻击的考验，逐渐被普遍认为是目前最优秀的公钥方案之一。RSA 的安全性依赖于大数的因子分解，虽然目前并没有从理论上证明破译 RSA 的难度与大数分解难度等价，不过这并不影响 RSA 的流行。RSA 可用于数字签名，具体操作时考虑到安全性和信息量较大等因素，一般可先进行哈希运算。由于进行的都是大数计算，使得 RSA 最大的问题是运行时间较长。无论是软件还是硬件实现，RSA 的运行速度一直不能和 DES 相比，一般来说只用于少量数据加密情况。

数字签名算法(DSA，Digital Signature Algorithm)也是一种非对称加密算法，被美国 NIST 作为数字签名标准(DSS，Digital Signature Standard)，DSA 一般应用于数字签名中。

#### 3. 单向加密

另一类算法是单向加密算法。该算法在加密过程中，输入明文后由系统直接经过加密算法处理，得到密文，不需要使用密钥。既然没有密钥，那就无法通过密文恢复为明文。

那么这种方法有什么应用呢？主要是可以用于进行某些信息的鉴别。在鉴别时，重新输入明文，并经过同样的加密算法进行加密处理，得到密文，然后看这个密文是否和以前得到的密文相同，从而判断输入的明文是否和以前的明文相同。

该方法计算复杂，通常只在数据量不大的情形下使用，如计算机系统口令保护措施中，这种加密算法就得到了广泛应用。近年来，单向加密的应用领域正在逐渐扩大。应用较多的单向加密算法有：RSA 公司发明的 MD5 算法；美国国家安全局(NSA)设计、美国国家标准与技术研究院(NIST)发布的 SHA 等。

MD5 的全称是 Message Digest Algorithm 5(消息摘要算法 5)，最初的目标是用于确保信息传输过程中的完整一致性。MD5 算法由 MD2 和 MD4 发展而来，其基本思想是：将大容量信息变换成一个定长的大整数，这个大整数对这个信息来说是单向的。一般来说，这个大整数称为消息摘要。MD5 能够获得随机长度的信息，然后生成 128 位的消息摘要。MD5 广泛用于密码认证、软件序列号等领域。

安全散列算法(SHA, Secure Hash Algorithm)是 NIST 发布的一个标准算法，一般称为 SHA-1，该算法也是一种单向加密算法，输入长度不超过 264 二进制位的消息，产生 160 位的消息摘要输出。消息验证码和 MD5/SHA1 算法的不同之处是：在生成摘要时，发送者和接收者都拥有一个共同的密钥。这个密钥可以是通过对称密码体系生成的，事先被双方共有，在生成消息验证码时，还必须有密钥的参与。只有同样的密钥才能生成同样的消息验证码。

### 3.1.3　密钥安全

对称和非对称加密系统的安全工作依赖于两个方面：加密算法和密钥。一般情况下，加密算法都是公开的，所以加密系统的安全性依赖于密钥的安全。一般说来，密钥应满足以下特性：不同情况下生成的密钥应是独立的、互不相关的，即每次生成的密钥和其他密钥无关；密钥值应是不可预测的；密钥值在某个范围内实现均匀分布。

实现密钥之所以具有以上特点，随机数起到了很大的作用。随机数生成是许多加密操作不可分割的组成部分，常被用于密钥的生成。同理，随机数也应有三个特性：均匀分布；数值不可预测；互不相关。如不具有这几个特性，随机数就被认为是不良的，对系统的安全性会产生巨大影响。产生随机数，有多种不同的方法，这些方法被称为随机数生成器，基本上

所有的高级语言都封装了随机数生成器。

密钥管理是一件很复杂的事情，从密钥的生成到密钥的销毁等各个方面都需要考虑。主要表现于：密钥的管理体制；密钥的管理协议；密钥生成、分配、销毁等。由于不同的加密体系，密钥生成之后的使用方法各不相同。如对称加密体系中，双方密钥必须相同；非对称加密体系中，公钥私钥成对出现。

对称加密体系中的密钥管理。对称加密体系中，采用对称加密技术的通信双方必须保证采用的是相同的密钥，由于密钥双方都需要知道，因此只能通过秘密方法传送。这就为密钥的传递带来了风险，必须保证彼此密钥的交换是安全可靠的，主要是防止密钥泄露或被敌方更改。因此，对称密钥的管理和分发工作是一件很有风险的事情。

解决这个问题的方法一般是：通过非对称密码体系来对对称密钥进行管理(采用该方法的一个原因是由于非对称密码体系适合对少量数据进行加密解密)。具体过程如下(为描述简便，此处只涉及密钥，没有涉及被加密的数据信息，实际上，被加密的数据信息也存在于通信过程中)：发送方生成对称密钥，将其用接收方的公开密钥加密，发出；接收方用自己的私钥将加密后的对称密钥解密，得到对称密钥。由于对信息的每次发送和接收，都对应了唯一的对称密钥，因此双方不需要对密钥进行维护。另外，即使泄露了密钥，敌方也无法知道密钥的内容，因为他不知道接收方的私人密钥，无法对对称密钥进行解密。这种方式使得管理相对简单和安全，同时，该方法还解决了对称密钥中存在的密钥篡改问题。

非对称加密体系中的密钥管理主要涉及的是公开密钥管理。一般情况下，通信双方间可以使用数字证书(公开密钥证书)来交换公开密钥。

国际电信联盟(ITU)制定的标准 X.509，对数字证书进行了定义，利用数字证书，可以确定如下内容：证书所有者名称；证书发布者的名称；证书所有者的公开密钥；证书发布者的数字签名；证书的有效期及证书的序列号等。证书发布者一般都是证书管理机构(CA)，它是通信各方都信赖的机构。

近年来，还出现了一些密钥管理芯片。密钥管理芯片是专门为密钥管理而设计的新一代加密芯片。可用于消费类电子产品，如视频处理板卡的数据流加密解密、游戏机板、路由器、机顶盒等。由于其加密解密速度较快，也得到了较广泛的应用。

## 3.2　认证及身份验证技术

信息网络中相互通信的两个实体往往物理上相隔很远，甚至从未谋面，那么一个实体如何确定是否真的在和另一个它所期望的实体通信？这正是认证及身份验证技术所要解决的问题。面对恶意的主动入侵者，鉴别远程实体的身份是十分困难的，密码学通常能为认证及身份验证提供良好的安全保证。

### 3.2.1　身份与认证

身份是实体在信息系统中的一种表示，用于区分不同的实体。实体指定唯一的身份表示。认证就是将某个身份与某个实体进行绑定。基于网络的认证机制要求实体向某个单一的系统进行认证，这个系统可以是本地的也可以是远程的。认证具有传播性。认证技术的共性是对

某些参数的有效性进行检验，即检验这些参数是否满足某种预先确定的关系。身份验证是应用系统安全的第一道关口，是所有安全的基础。

在信息安全系统中，某个身份就对应某个用户。一个身份可以是一个任意个数的包含字母和数字的字符串表示的名字，它可能在某些方面是受限制的(如访问控制)。身份可以指由多个实体组成的主体，即群组。群组是可以快速对实体集执行访问控制和其他安全策略的一种简便方法，可作为把实体关联起来的基础。群组模型有静态模型和动态模型。如 Alice 属于某个实体集合就是静态模型，动态模型将实体集动态组建成分组。某个身份可能对应着一个角色集合。例如，当 UNIX 用户登录后，它们被分配到一个群组成为该群组的成员。用户参与的每个进程都具有两种身份，即用户身份和群组身份。

身份鉴别是向信息安全系统表明某个身份的过程，是通过将证据与实体身份绑定来实现的。证据与身份之间是一一对应的关系。双方通信过程中，一方实体向另一方实体提供证据证明自己的身份，另一方通过相应机制来验证证据，以确定该实体是否与证据所宣称的身份一致。

当用户登录计算机、自动柜员机、电话银行系统或其他通信终端时，如何确认该用户是谁呢？认证可以确认用户的身份，可以防止恶意用户对信息的主动攻击。

身份验证就是某个实体证明他/她就是他/她所说的某个身份的过程。通过认证可将一个实体绑定为信息安全系统内部的一个身份。认证与身份鉴别的区别在于，认证协议中 Alice 可向 Bob 证明她是 Alice，但是任何其他人都无法向 Bob 证明她也是 Alice，即其他人不能在 Bob 面前冒充 Alice；身份鉴别协议中 Alice 可以向 Bob 证明她是 Alice，但是 Bob 无法从中得到额外的信息，以便向其他人证明他也是 Alice，即 Bob 不能在其他人面前冒充 Alice。

认证方法主要有 4 种：实体知道什么，如身份证号码、个人识别码(PIN，Personal Identification Number)、出生日期(DoB，Date of Birth)等。实体拥有什么，如证章、信用卡、ID 卡和密钥等。实体是什么，如指纹、声音、虹膜等。实体在哪儿，如特定的大门、特殊的终端、特别的访问设备等。

单向认证是 A、B 双方在网上通信时，B 只需要认证 A 的身份即可。在简单的认证协议中，往往一方是主动提问并验证对方的身份，而另一方则处于被动的地位接受检验，许多访问控制所用的认证协议就属于此类。但在很多网络应用场合，通信双方实际上是完全对等的实体，他们同样有权要求验证对方的身份以维护自身的利益。这就是所谓的双向认证问题。

双向认证是 A、B 双方在网上通信时，B 不但要认证 A 的身份，A 也要认证 B 的身份。双向认证不是两个单向认证协议的简单重复。如果所用的加密算法和密钥是安全的，那么相应的协议可以认为是安全的。但如果将质询应答认证协议用两次，用于双向认证时就却很容易被攻击者找到漏洞。

可信第三方认证也是一种通信双方互相认证的方式，但认证过程必须借助于一个双方都信任的可信第三方。当双方欲进行通信时，彼此必须先通过可信第三方的认证，相互交换密钥，然后通信。由这种借助于可信第三方的认证方法变化而来的认证协议相当多，其中典型的例子就是 Kerberos 认证协议。

### 3.2.2　身份验证技术

身份验证一般涉及识别和验证两个方面的内容。识别是指要明确访问者是谁，即必须对

系统中的每个合法用户都有识别能力。要保证识别的有效性，必须保证任意两个不同的用户都不能具有相同的识别符。所谓验证是指在实体声称自己的身份后，信息安全系统对他所声称的身份进行确认，以防假冒。识别信息一般是非秘密的，而验证信息必须是保密的。身份验证技术是指计算机及网络系统确认操作者身份的过程所应用的技术手段，身份验证技术有很多形式，包括口令、质询应答认证协议、利用信物的认证技术、生物认证技术等。

### 1. 口令

口令是与特定实体相关联的信息，用来证明实体所声称的身份确实属于该实体。基于口令的认证方法属于“实体知道什么”。实体提供一个口令，安全系统检查这个口令的有效性。如果这个口令和这个实体相关联，那么这个实体的身份被认证通过了。否则，这个实体被拒绝，认证失败。

另一种与口令类似，根据“实体知道什么”进行身份验证的方法是：当某用户第一次进入系统时，系统向他提出一系列问题，如他曾就读的高中全称、他父母的血型、他喜欢的作者名字以及他喜欢的颜色等。不是所有问题都必须回答，但要回答足够多的问题。有些系统还允许用户添加自己定义的一些问题与答案。系统要记住用户的问题与相应的答案，以后当该用户再访问系统时，系统就要向他提出这些问题，只要他能够正确地回答出足够多的问题，系统就认为该用户具有他所声称的合法身份。这种方法的优点是对用户比较友好，用户可以选择他非常熟悉而其他人又不容易获得正确答案的信息作为问题，所以安全性是有一定保障的。缺点是系统与用户之间需要交换的认证信息较多，有时觉得不方便或者比较麻烦。另外，这种方法还需要在系统中占据较大的存储空间来存储认证信息，认证时间也相应长一些。安全性完全取决于对手对用户的背景知道多少，所以在高度安全的系统中，这种方法仍不适用。

对口令的最常见攻击就是字典攻击。字典攻击通过重复试验和连续排错的方法来猜测口令。常见的字典攻击有两种类型：一是如果补充信息 C 和补充函数 F 可以获得，那么针对每一个猜测 p 和每一个计算 f(p)，直到计算的结果与保存的同一实体的补充信息 C 相匹配为止；二是如果验证身份的认证函数集合 L 可以获得，那么针对每一个猜测 p 和每一个计算 l(p)，如果计算结果为真，则 p 就是正确的口令。对抗口令猜测的目的是最大限度地增加攻击者猜测出正确口令所费的时间。口令猜测攻击是不能防止的，因为认证函数必须公开可用，使合法用户可以访问系统。除了通过验证口令，系统无法区别授权用户与非法用户。

口令验证是根据用户知道什么来进行的。口令验证方法已经广泛应用于日常生活的各个方面，从阿里巴巴四十大盗的开门咒语到军事领域的哨兵口令，以及目前在计算机系统中的注册口令。在这种方法中，人们主要关注的是口令的生成与管理。

目前口令生成主要有两种方法：一种是由口令拥有者自己选择口令；另一种是由计算机自动生成随机的口令。前者的优点是用户很容易记忆，一般不会忘记，缺点是易于猜测；后者的优点是随机性好，难于猜测，缺点是用户记忆困难。用户自己选择的口令大多是用户的姓名、街道名、城市名、汽车牌照、房间号码、手机号码、倒过来拼写的有意义的字等。对于想要窃取他人口令的人来说，这些都是要优先猜测的目标。

用户可选择容易记忆的口令，但要拒绝容易被猜到的口令。容易猜测的口令有以下一些类型，用户应该避免。如基于账号名的口令；词典里的词；翻转字典里的单词；把字典单词

的其中一部分或全部字母大写；将字典里词中的任意字母替换为控制字符；对字典里的词进行简单变换；对字典里的词进行动词变化或词形变换；短于 6 个字母的口令；仅包含大写或小写字母，或者字母和数字，或者字母和标点符号；像许可证号码的口令；首字母简略词；过去用过的口令；字典单词的连接；在字典单词的前面或后面添加一些数字、标点符号或空格；把字典单词所有的元音字母删除；把字典单词中的空格删除等。

对口令的管理至关重要，口令在系统中的保存就是一个问题。对于采用口令方法来认证用户身份的信息系统来说，如果同时有许多用户在其中注册，那么相应地，每个用户都要有自己的口令，并且原则上，不同用户的口令是不同的。口令要严格保密，不能被其他用户得到(不管用什么方法)。系统要想对用户的身份进行认证，就必须保存用户的口令，但是口令显然不能以明文形式存放在系统中，否则口令很容易泄露。如果采用通常的加密方法对存放在系统中的口令进行加密(如 DES 算法)，那么加密密钥的保存就成了验证的安全问题，一旦加密密钥泄露出去，就可能把系统中所有的口令都泄露出去。所以，口令在系统中的保存应该满足如下要求，即利用密文形式的口令恢复出明文形式的口令在计算上是不可能的。口令一旦加密，就永远不会以明文形式在任何地方出现。这就是说，要求对口令进行加密的算法是单向的，只能进行加密，解密是不可能的。系统利用这种方法对口令进行验证时，首先对用户输入的口令进行加密运算，将运算结果与系统中保存的该口令的密文进行比较，相等即为合法，不相等即为非法。

口令管理的第二个问题是关于口令的传送问题。口令一定要以安全方式传送，否则就可能泄露使其失去意义。用加密方法解决不了这个问题，因为即使采用加密方法，也必须对接收者的身份进行认证，如果对接收者的身份不加认证，就无法保证口令会正确地传送到合法用户手中。而对接收者的身份进行认证是口令要解决的问题。所以，在口令建立起来之前无法对接收者的身份进行认证，也就无法保证口令能传送到正确用户的手中，必须考虑采用其他方法。通常采用寄信的方式，许多银行就是利用这种方法向它们的客户分送个人 PIN 码的。银行系统通常采用夹层信封，由计算机将口令印在中间纸层上，外边看不到，只有拆封才能读出。若用户收到的信封已被拆阅，可向银行声明拒用此口令。

当用户进入系统时，诸如某些与真实网页有相似域名并且版面看起来很像原网页的虚假网页，用户终端屏幕上会出现这样的请求“请输入口令”。假如这时用户未能发现域名相差一个字符，就会不假思索地打入他的口令，但这很可能是一个骗局。由于系统没有向用户证明它是真正、正确的系统，用户面对的可能是一个专门设计的用于窃取用户口令的假冒网页。

### 2. 质询应答协议

传统的身份验证机制是建立在静态口令的识别基础之上的，这种以静态口令为基础的常规身份验证方式存在许多口令被窃取的隐患。包括以下几种：网络数据流窃听，很多通过网络传递的认证信息是未经加密的明文(如 FTP、TELNET 等)，容易被攻击者通过窃听网络数据分辨出认证数据，并提取用户名和口令；认证信息截取、重放，简单加密后进行传输的认证信息，攻击者会使用截取、重放方式推算出密码；字典攻击，以有意义的单词或者数字作为密码，攻击者会使用字典中的单词来尝试用户的密码；穷举尝试，这是一种特殊的字典攻击，它使用字符串的全集作为字典。如果用户的密码较短，容易被穷举方法破解。

随着网络应用的深入化和网络攻击手段的多样化，静态口令认证技术由于其自身的安全

缺陷，已不再适用于安全性要求较高的网络应用系统。针对静态口令认证技术存在的安全缺陷，业界提出了一次性口令身份验证技术(One-Time Password Authentication)，也称为动态口令认证技术。动态口令认证就是在登录过程中加入不确定因素，使每次登录时传送的认证信息都不相同，以提高登录过程的安全性。一次性口令是质询应答协议采用的方法之一，一个口令一旦使用就失效了。质询应答协议采用的方法之二是硬件支持的质询应答程序，根据输入质询，硬件设备计算出一个适当的应答。

质询应答协议旨在对抗重放攻击。在质询应答协议的每一步均有一个历史时间戳，保证了协议可有效地抵抗重放攻击。质询应答协议消除了传统口令认证技术的大部分安全缺陷，能有效抵抗传统口令认证技术所面临的主要安全威胁和攻击，为网络应用系统提供了更安全可靠的用户身份验证保障。

静态口令存在安全隐患，在 20 世纪 80 年代初，一次性口令的思想被首次提出。一次性口令即一个口令只对一次使用有效，被使用之后就变为无效口令，这是口令时效性的一种极端形式。在某些场合，质询应答机制使用一次性口令。考虑把应答作为口令，由于连续认证的质询是不同的，因此应答也是不同的。因此，每个应答在被使用后就变为无效。

一次性口令的基本思想是：用户每次同服务器连接过程中使用的口令在网上传输时都是加密的密文，而且这些密文在每次连接时都是不同的，也就是说密文是一次有效的。当一个用户在服务器上首次注册时，系统给用户分配一个种子值(seed)和一个迭代值(iteration)，这两个值构成了一个原始口令。同时，在服务器端保留只有用户自己知道的口令，口令由数字、字母、特殊字符、控制字符等组成的长为 58 个字符的字符串组成。用户每次向服务器发出连接请求时，服务器把用户的原始口令传给用户。用户接到原始口令后，利用口令生成程序，采用散列算法(如 MD5)，结合口令计算出本次连接实际使用的口令，再把口令传回服务器。服务器保留用户传来的口令，然后调用口令生成器，采用同一散列算法，利用保存在服务器端的口令和刚传给用户的原始口令自行计算生成一个口令。服务器把这个口令与用户传来的口令进行比较，进而对用户进行身份确认。每次身份成功认证后，原始口令中的迭代值自动减 1。该机制由于每次登录时的口令是随机变化的，每个口令只能使用一次，彻底防止了前面提到的各种窃听、重放、假冒、猜测等攻击方式。

使用硬件支持一次性口令就相对简单，因为口令不需要打印在纸上或者输出到某些中间媒体。硬件支持的质询-应答程序有两种形式：通用计算机和专门硬件。两者实现相同的功能。

第一种类型的硬件有一个非正式的名称 token，它提供了一种对消息进行 Hash 和加密的机制。使用这种设备时，系统首先发送一个质询。用户把这个质询输入到设备里，设备返回一个适当的应答。一些设备要求用户输入个人口令，并把这作为密钥，或者结合质询一起用来产生应答。

第二种类型的硬件基于时间。每隔 60 秒，就显示一个不同数字，这些数字的变化范围可由系统决定。使用这种设备时，用户首先提交登录名，系统请求一个口令，接着用户输入硬件显示的数字，后面跟一个固定的口令。系统验证数字是系统在当时期望用户提交的数字，同时固定口令正确。

### 3. 信物身份验证

对大多数人来说，利用授权用户所拥有的某种东西进行访问控制的方法并不陌生，我们

经常使用这种方法。在日常生活中，几乎所有人都有钥匙，有的用于开房门，有的用于开抽屉，有的用于开车子。对信息系统的访问控制也可以利用这种方法。可在信息系统终端上加一把锁，使用该终端的第一步就是用钥匙打开相应的锁，然后再进行相应的注册工作。但对信息系统来讲，这种方法的最大缺点是它的可复制性。我们所用的普通钥匙是可以任意复制的，并且也很容易被人偷去。针对这个问题，人们找到多种解决方案。

磁卡是目前广泛使用的一种设备，它是一个具有磁性条纹的塑料卡，这种卡已经广泛用于身份识别，如信用卡、校园一卡通、第二代身份证、公交卡，以及对安全区域的访问控制等。国际标准化组织(ISO)发布了相关标准，对卡的尺寸、磁条的大小等都有具体的规定，该组织还制定了其他几个标准，对相应的数据记录格式也做了规定。

磁卡中最重要的部分是磁条，磁条中不仅存储着数据，而且存储着用户的身份信息。在美国，磁卡一般与个人识别号 PIN 一起使用。PIN 是用于保护磁卡免受误用的秘密识别代码。PIN 与密码类似，只有磁卡的所有者才知道该 PIN。只有拥有该磁卡并知道 PIN 的人才能使用该磁卡。在脱机系统中，PIN 必须以加密形式存储在磁卡中。识别设备首先读出该卡中的身份信息，然后将其中的 PIN 解密，并要求用户输入 PIN，识别设备将这两个 PIN 进行比较以决定该卡的持有者是否合法。在信息系统中，PIN 可以不存在卡上，而存在系统中，进行认证时，系统把用户输入的 PIN 与系统中的 PIN 进行比较，据此来判断该卡的持有者是否具有他所声称的身份。

目前，人们常用的是智能卡。这种卡与普通磁卡的区别在于，这种卡带有智能化的微处理器与存储器，具有更高的防伪能力，一般不易伪造，因而更加安全。智能卡已经广泛应用于我国银行、电信、交通等社会的各个方面，非接触式智能卡已被证明是处理大量交易最有效率的工具，最明显的例子是市政交通一卡通。市民持市政交通一卡通卡即可去部分便利店、超市、西饼屋、餐厅、药店、电影院刷卡消费。市民只要拥有一张市政交通一卡通卡，就可以实现日常生活中的交通出行以及日常生活的一卡付费，从而为市民提供了一种真正便捷的付费方式。

在基于证书的认证系统中，对私钥的保护是极其重要的。一个设计的更好的系统会将私钥保护起来，并将它与计算机隔开。智能卡就可以做到这一点。尽管有许多种类型的智能卡，而用于认证的智能卡看起来就像信用卡，但它包含了一块用来保存私钥和证书副本的计算机芯片，并且能进行相关处理。在为特定的应用选择合适的智能卡时，必须特别注意它们的应用场合。一些额外的硬件令牌，可使用基于 USB 的接口来实现类似的用途。智能卡需要特殊的智能卡读卡器来提供智能卡和计算机系统之间的通信。

#### 4. 生物认证

前面讨论了利用口令与信物进行身份验证的方法，由于口令可能会被不经意地泄露，而信物又可能丢失或者被人伪造，所以，在对安全性要求较高的情况下，这两种方法都不太恰当。为此，人们把注意力集中在利用人类特征进行认证的方法。

人类特征可以分为两种：一种是人的生物特征；另一种是人的行为特征。使用生物特征作为身份证明与人类历史一样悠久。通过声音、外表来识别一个人，通过易容术来冒充一个人，这在古代就被广泛使用，这一点在金庸先生的小说中也有类似的描述。利用人类的生物特征进行身份识别的历史已经很长了，特别是在侦破犯罪案件中。法国在 1870 年前的四十多

年中，一直使用一种称为 Bertillon 的系统，它通过测量人体各部分的尺寸来识别不同的罪犯，如前臂长度、各手指长度、身高、头的宽度、脚的长度等。

生物认证就是通过生物学特征和行为特征来辨识每个人的自动化方法。常用特征有指纹、声音、眼睛、脸部或者以上特征的综合。当给定用户一个账号，系统管理员要采用一系列措施，在一个可接受的错误程度内识别该用户。只要该用户访问系统，生物认证机制就要验证用户的身份。只要与声明用户身份关联的已知数据进行比较，就可以确定该用户是得到认证还是被拒绝。人的特征具有很高的个体性，世界上几乎没有任何两个人的特征是完全相同的，所以这种方法的安全性极高，几乎不能伪造，对于不经意的使用也没有什么副作用，但一般来讲，采用这种方法费用都很昂贵。

人的下意识动作也会留下一定特征，不同的人对同一个动作会留下不同的特征，这方面最常见的例子是手写签名。手写签名是一种历史悠久的身份验证的方法。商人之间签订合同、政府间签署协议、某组织下发文件等活动都需要有相应负责人的签字，以表明签字人对文件的认可。频繁进行签名的人对这一动作已经司空见惯，所以，签名已经成了一种条件反射的动作，它已经不受手臂肌肉的人为控制，从而成为一种下意识的动作。这种动作的结果会留下许多特征，如书写时的用力程度、笔记的特点等，根据这些特征就能认证出签名人的身份。

## 3.3　访问控制技术

访问控制是在保障授权用户能获取所需资源的同时，拒绝未授权用户的安全机制，是信息安全的重要组成部分。在认证和授权后，访问控制机制将根据预先制定的规则对用户访问的资源进行控制，只有规则允许的资源才能访问。访问方式可以是获取信息、修改信息或者完成某种功能，一般是读、写或执行。

### 3.3.1　访问控制和安全机制的设计原则

#### 1. 访问控制技术

访问控制是指主体依据某些控制策略或者权限对客体或其资源进行的不同授权访问。访问控制包括三个要素：主体(Subject)、客体(Object)和控制策略。控制策略是主体对客体的访问规则集。规则集定义了主体对客体的作用行为和客体对主体的条件约束。访问控制策略体现了一种授权行为，也就是客体对主体的权限允许，这种允许不能超越规则集。

在讨论信息保护问题时，从概念上说，可为每个需要保护的客体建立一个坚固的保护墙。保护墙上留有一个门，门前有一名门卫，所有对客体的访问都首先在门前接受门卫的检查。在整个系统中，有很多客体，因而有很多保护墙和门卫。对客体的访问控制机制的实现可分为两种类型：面向标签(Ticket-Oriented)的实现和面向名单(List-Oriented)的实现。在面向标签的实现中，门卫手中持有一份对一个客体的描述。在访问活动中，主体携带一张标签，标签上有一个客体的标识和可以访问的方式，门卫把主体所持标签中的客体标识与自己手中的客体标识进行对比，以确定是否允许访问。在整个系统中，一个主体可能持有多张标签。在面向名单的实现中，门卫手中持有一份所有授权主体的名单及相应的访问方式，在访问活动中，

主体出示自己的身份标识，门卫从名单中进行查找，检查主体是否记录在名单上，以确定是否允许访问。

访问控制的目标分为以下几个方面：

- 机密性要求，防止信息泄露给未授权的用户。一般来说，系统中的某些信息非常重要，如公司的财务信息以及个人用户的信用卡账号等。对于这些信息来说，任何未经授权的信息泄露都会给企业和个人带来损失。
- 完整性要求，防止未授权的用户对信息的修改。完整性要求是为了保证系统资源处于一个有效的、符合预期的状态，防止资源被不正确或不适当地修改。同时，也为了维护系统不同部分的一致性。
- 可审计性要求，防止用户对访问过某信息或执行过某一操作进行否认。
- 可用性要求，保证授权用户对系统信息的可访问性。可用性是为了保证系统顺利工作，即保证已经获得授权的用户对系统信息的可访问性。当系统可用性要求有冲突时，必须创建新的安全规则。如果系统的安全性很好，但某些情况下不可用，那么这个系统也不是一个成功的系统。

### 2. 安全机制设计原则

设计与实现安全机制的基本原则包括以下 8 个原则：

- 最小权限原则，分配给系统中的每个程序和每个用户的权限应该是它们完成工作所必须享有的权限的最小集合，这和“需要知道”原则类似。如果主体不需要访问特定客体，则主体就不应该拥有访问这个客体的权限。
- 自动防故障默认原则，访问判定应建立在显式授权而不是隐式授权的基础上。显式授权指定的是主体应该拥有的权限，隐式授权指定的是主体不应该拥有的权限。默认情况下，没有明确授权的访问方式，应该视为不允许的访问方式。.
- 权限分离原则，为一项权限划分出多个决定因素，仅当所有决定因素均具备时，才能行使该项权限。例如，一个档案柜设有两把钥匙，由两个人掌管，仅当两个人都提供钥匙时，档案柜才能打开。如果其中任何一人不提供钥匙，都无法打开档案柜。同样，系统在对资源访问授权时也至少应该满足两个条件。这种方法提供了一种资源的细粒度访问控制，也保证了访问的合法性。
- 完全仲裁原则，对每个客体的每次访问都必须经过检查，以确认是否已经得到授权。只要主体发出对某客体的访问请求，操作系统都对该操作进行仲裁。操作系统需要检查该主体是否被允许访问该客体。如果允许，操作系统将提供访问所需的资源。如果这个主体再次请求对该客体的访问，操作系统必须再次检查这个主体是否还被允许执行这种访问。
- 开放式设计原则，不应该把保护机制的抗攻击能力建立在设计保密的基础之上，应该在设计公开的环境中设法增强保护机制的防御能力。
- 最小公共机制原则，尽量减少由两个以上用户共用和被所有用户依赖的机制数量。每个共享机制都是一条潜在的用户间的信息通路，要谨慎设计，避免无意中破坏安全性。应证明为所有用户服务的机制能满足每个用户的要求。
- 机制经济性原则，保护机制应设计得尽可能简单短小。如果设计和实现机制简单，

则存在错误的可能性就小，进程的测试也就简单。有些设计和实现中的错误可能产生意想不到的结果，而这些错误在常规使用中察觉不到。简单短小的设计是这类工作成功的关键。

- 心理可接受性原则，为使用用户方便自如地正确运用保护机制，用户界面应设计得便于使用。可以理解为安全机制可能给系统增加了额外负担，但这种负担必须是最小的而且合理。

### 3.3.2　访问控制列表

访问控制列表(ACL，Access Control List)是以文件为中心建立的访问权限表。目前，大多数 PC、服务器和主机都使用 ACL 作为访问控制的实现机制。ACL 的优点是实现简单，对系统性能影响较小。

从概念上说，访问控制可以通过使用和维护一个访问控制矩阵(ACM, Access Control Matrix)来实现。访问控制矩阵是通过矩阵形式表示访问控制规则和授权用户权限的方法。访问控制矩阵是二维的，包含三个元素：主体、客体和访问权限。对主体而言，拥有对哪些客体的哪些访问权限，而对客体而言，有哪些主体可以对它实施访问。将这种关联关系加以描述，就形成了访问控制矩阵。其中，特权用户或特权用户组可修改主体的访问控制权限。访问控制矩阵的行对应于主体，列对应于客体。第 i 行第 j 列的元素是访问权限的集合，列出了允许主体对客体进行访问的权限。

如果系统中用户和资源都非常多，而每个用户可能访问的资源有限，将出现庞大的访问控制矩阵中存在很多空值的情况，从而造成矩阵的存储空间的浪费。简单的解决方式是将访问控制矩阵按行或按列进行划分。如果按行划分，得到访问控制能力表。即对于每个用户，能力表列出可以使用的客体和对客体的访问能力。如果按列进行划分，可得到广泛应用的访问控制列表。即对于每个资源，访问控制列表中列出可能的用户和用户的访问权限。

访问控制列表是基于访问控制矩阵中列的自主访问控制区，它在一个客体上附加一个主体明细表，来表示各个主体对这个客体的访问权限，明细表中的每一项都包括主体的身份和主体对这个客体的访问权限。访问控制列表机制适合于少数需要被区分的用户，并且这些用户大都比较稳定。如果访问控制列表太大或经常改变，那么维护访问控制列表就成为一个问题。

#### 1. Windows NT 访问控制列表

Windows NT 系统中共享对象的访问权限是由对象所有者决定的。如果用户想共享某个对象，他首先要为该对象选择唯一的名字，然后为其他用户和组分配访问权限。Windows NT 允许用户或组对文件或目录执行以下 6 种操作：读、写、执行、删除、改变许可和获取拥有权限，这些权限称为普通权限。

Windows NT 文件的普通权限分为以下几类：no access，主体不能访问该文件；read，主体能读或执行该文件；change，主体能读、执行、写或删除该文件；full control，主体拥有该文件的所有权限；special access，允许分配任意权限；special access，允许分配任意权限的组合。

Windows NT 目录的普通权限分为以下几类：no access，主体不能访问该目录；read，主体能读或执行该目录中的文件；list，主体能列出该目录中的内容，并且也可转入该目录下的子目录；add，主体能在该目录下创建文件或子目录；add and read，是普通权限 add 和 read

的结合；change，主体能创建、读、执行或写该目录下的文件，也能删除子目录；full control，主体拥有该目录下的文件及其子目录的所有权限；special access，允许分配任意权限的组合。

当用户访问文件时，Windows NT系统首先检查文件的ACL。如果该用户没有出现在ACL中，并且不是ACL所列组中的任一成员，则访问被拒绝。否则，如果任一ACL条目拒绝用户的访问，系统就拒绝访问(这是一种显式拒绝)。如果访问不被显式拒绝，而用户又在ACL中出现，则该用户的权限集就是所有指定此用户的ACL条目中权限集的并集。

#### 2. UNIX 访问控制列表

大多数UNIX操作系统都支持访问控制列表。UNIX访问控制列表中包括三类主体：文件的所有者、组和其他用户；有三种访问权限：读(r, read)，写(w, write)和执行(x, execute)。访问控制列表用9位许可字段表示：rwxrwxrwx。其中，前3位字段表示所有者的访问权限，中间3位字段表示组的访问权限，后3位字段表示其他用户的访问权限。例如，rw-r-----表示所有者有读、写的权限，组成员有读的权限，其他用户没有访问权限。可见，任何用户和组都可以通过3个保护位(读、写、执行)与文件相关联，这提供了一种分配访问权限的灵活机制。

### 3.3.3　能力表

能力是访问控制中的一个重要概念，它指访问请求者所拥有的一个有效标签，授权标签表明的持有者可按何种方式访问特定的客体。用二元组(x, y)表示对一个客体的访问能力，其中，x是该客体的唯一的名字，y是对该客体x的一组访问权限的集合。访问控制能力表是以用户为中心建立访问权限表。

能力是为主体提供的对客体具有特定访问权限的标志，它决定主体能否访问客体以及以什么方式访问客体。主体可将能力转移给为自己工作的进程，在进程运行期间，还可添加或者修改能力。

能力和自主访问控制相联系，能力表是最常用的基于行的自主访问控制。当主体创建新的对象时，它可以通过授予其他主体合适的能力以允许其他主体访问这个对象。同样，当一个主体调用另一个主体时，该主体可将它的能力或部分能力传递给被调用的主体。

根据每个主体的能力表，可决定主体是否可对给定的客体进行访问以及进行哪些访问。能力表在时间效率和空间效率上都优于访问控制矩阵，但能力表也有自身的缺点。对于一个给定客体，要确定所有有权访问它的主体，用户生成一个新的客体并对其授权或删除一个客体时都比较复杂。

如果主体拥有某一对象的能力，那么该主体可访问这一对象。能力本身就是被保护的对象，用户必须保护它们不被修改。有三种方法可用于保护能力表：标签、受保护内存和加密技术。

1) 标签，标签式结构中有一个与每个硬件字相关的比特集合。标签有set和unset两种状态。如果标签的状态是set，则普通进程就可以读这个字，但不能更改。如果标签状态是unset，则普通进程就可以读和修改。普通进程不能修改标签的状态，只有处于特权模式下的处理器才能进行修改。例如：B5700系统使用了标签式结构。标签域包含三个比特，指示这种结构如何使用这个字(指针、描述符、类型等)。

2) 受保护内存，该方法是使用与内存分页或分段相关联的保护比特。所有的能力表都存

储在一个内存页面中，进程只能读取但不能改变这些内存。例如：CAP 系统不允许进程修改存放指令的内存分段，它也将能力表存放在这个分段。一个防护寄存器将指令与能力表隔开。

3) 加密技术，密码校验和是实现信息完整性检验的一种机制。每个能力表都有一个与之相关的密码校验和，该校验和由密码系统进行加密，而操作系统掌握密钥。当进程向操作系统提交能力表时，系统首先重新计算与能力表关联的密码校验和。然后，系统可使用密钥加密该校验和，并将结果与能力表中存储的校验和进行比较，或者解密能力表中的校验和，并与计算得出的校验和相比较。如果结果匹配，那么能力表没有被修改，否则，能力表被拒绝。

在能力表中，如果撤消对一个客体的访问权限，需要撤消所有对该客体授权的能力表。理论上，要求对每个进程进行检查，删除相关能力表。但这种操作开销过大，所以要使用其他替代方法。

最简单的机制是间接客体引用。定义一个或多个全局客体表，在此模式下，每个客体在表中有一对应的条目。能力表不直接指定客体名字，而是指定客体在全局客体表中的条目。

该方法具有以下优点：要撤消客体权限，只需将该客体在全局客体表中的条目设为无效，这样所有对该客体的引用都将返回一条无效条目，访问被拒绝。如果只撤消客体的部分权限，该客体可以有多个条目，每个条目对应不同的访问权限集或不同的用户组。例如：Amoeba 系统使用的是这种方案。要撤消一个能力表，客体的拥有者请求服务器改变能力表的随机数，并发布新的能力表。这样可将现有的能力表置为无效。

另一种撤消权限的机制是使用抽象数据类型管理器。每种抽象数据类型中包含一个权限撤消子程序。当访问权限被撤消时，类型管理器只需禁止被收回权限的主体再次访问即可。

对于能力表来说，如果出示有效的能力，则允许访问，不会验证能力持有者的身份。能力表更适用于分布式环境。保护机制和命名可合为一体，使访问控制更加灵活。访问控制列表可以提供更好的保护，因为它总是在允许用户访问之前识别用户，也更容易跟踪使用对象的人。

### 3.3.4　锁与钥匙

锁与钥匙的技术结合了访问控制列表和能力表的特性。一部分信息(锁)与客体相关联，另一部分信息(钥匙)与授权访问该客体的主体以及允许主体访问客体的方式相关联。当主体打算访问客体时，就检查主体的钥匙集。如果主体有一把钥匙与客体的任一把锁相对应，就赋予主体正确的访问类型。

与其他访问控制机制不同，锁与钥匙具有动态的特点。访问控制列表是静态的，必须人为进行修改。而锁与钥匙会因响应系统的约束、增加条目的通用指令或其他任何因素而发生改变，不需要手工修改。

在锁与钥匙的密码学实现中，客体 o 由一个密钥加密，主体拥有解密密钥。为了访问客体，主体对客体进行解密。系统实现了一种名为 or-access 的方法，允许 n 个主体访问数据。该方法使用 n 个不同密钥对数据的 n 个副本进行加密，每个主体拥有一个密钥。系统也可以实现 and-access 方法，n 个主体的访问请求同时发生才不拒绝访问。使用 n 个密钥进行迭代加密，每个主体拥有一个密钥。

以 IBM370 系统为例，它为每个进程分配一个访问密钥，并为每个页面分配一个存储密钥及一个提取比特。如果提取比特被清零，则只允许读访问。如果提取比特置为 1 且访问密

钥是 0，该进程可对任意页面进行写操作。如果访问密钥不是 0，也不与存储密钥匹配，则进程不能访问该页面。

### 3.3.5　基于环的访问控制方法

Multics 系统是一个分时操作系统，该系统开始作为一个合资项目，是 1964 年由贝尔实验室、麻省理工学院及美国通用电气公司所共同参与研发的，旨在开发一套安装在大型主机上的多人多工操作系统。

Multics 系统推广超级用户与用户的状态表达，产生了一种称为基于环的访问控制的保护机制。保护的基本单元是内存或磁盘上的一个分段。分段有两类：数据和程序。每个分段可以有与其相关的读、写、执行和添加权限。这些权限包含在访问控制列表中，访问控制列表对每个用户的访问进行约束。

Multics 系统定义了一系列保护环，编号为 0 至 63。这些数字对应同心保护环，其中环 0 位于中心，提供最高级别的保护。环的编号越大，环中分段的特权越低。

根据下面的访问规则，程序能“跨越”环的边界。某些情况下，跨越环将产生“环跨越错”，并产生对内核的陷入。此时，gatekeeper 机制将检查参数，并访问、执行其他例程以限制环跨越。Gate 是一个入口点，它在程序中特别声明，编译器和链接器生成特殊代码，使得这些入口点可被其他程序使用。

假设在环 r 中执行的程序打算访问一个数据段，与每一个数据段关联的是一对环编号 ($a_1$，$a_2$)，称为访问等级，其中 $a_1 \leqslant a_2$。访问数据段的规则如下：当 $a_1 < r \leqslant a_2$ 时允许访问。当 $r \leqslant a_1$ 时允许读和执行，拒绝写和添加。当 $a_2 < r$ 时所有访问都拒绝。

假设该程序在环 r 中执行，打算访问一个程序段。每个程序分段有一个访问等级($a_1$，$a_2$)，和一个调用等级($a_3$，$a_3$)。访问程序段的规则如下：当 $r < a_1$ 时，允许访问，但会发生环跨越错。当 $a_1 \leqslant r \leqslant a_2$ 时，允许所有访问，且不会发生环跨越错。当 $a_3 < r$ 时，拒绝所有访问。

例如，假设一个数据分段具有访问等级(2, 4)，而 John 有对该数据分段的读写权限。如果 John 的程序在环 1 中执行，则它对此数据段的读操作将成功。如果 John 的程序在环 3 中执行，则任意读操作成功，而写操作失败。如果 John 的程序在环 5 中执行，则所有访问请求都将失败。

## 3.4　安全保障

在现实世界中，没有一个系统是绝对安全的，使用安全保障技术可使系统在安全性、可靠性和鲁棒性方面都得到增强。本章将具体描述安全保障的相关概念、基本思路和方法。

### 3.4.1　保障模型和方法

安全保障是判断信息系统可信度的基础，安全保障技术用来检验需求的正确性以及设计、实现和维护的有效性，能显著提高系统的可信度，及早发现错误并修正错误。在系统的生命周期中，从建立需求到系统设计，到开发，再到测试和维护，每个过程都应该采用相应的安全保障技术。好的生命周期模型不仅可以提高开发软件的质量，也可以增强其安全性。

### 1. 安全保障和信任

系统的安全性可能是由几个安全机制，或者是一组定义清晰的安全需求和满足安全需求的系统实现。然而，仅提供安全需求或者一些安全功能并不能使一个系统变成可信任系统。要证明一个系统是可信的，需要使用一些方法和尺度。如果有足够的可信事实证明一个系统满足一系列安全需求，则这个系统是可信任的。可信任系统是被证明满足指定安全需求的系统，是由一个被授权评估系统安全等级的专家组织来进行评估的。将一个系统可接受的信任程度加以分级，凡符合某些安全条件、基准、规则的系统即可归类为某种安全等级。将系统的安全性能由高到低划分为几个等级，并且较高的等级的安全范围涵盖较低等级的安全范围，也就是说，较高的等级比较低的等级有更严格的安全保障需求。

如果有足够可信的事实来证明一个系统满足一组安全需求，则这个系统是可信任的。信任是用来衡量一个系统的可信任程度的，它依赖于所能提供的可信事实。对信息系统的信任应该建立在系统的设计和实现满足安全需求的事实基础上。也就是说，可信任系统的基础是安全系统。

安全保障是通过使用多种多样的安全保障技术而获得的，这些安全保障技术提供证据来证明系统的实现和运行能够满足安全策略中定义的安全需求。安全保障技术提供证据的方法可分为非形式化方法、半形式化方法和形式化方法。

形式化方法是用一套特制的表意符号(其意义可以解释的)去表示概念、判断、推理，获得它们的形式结构，从而把对概念、判断、推理的研究，转化为对形式符号表达式系统的研究的方法。凡是采用严格的数学工具、具有精确数学语义的方法，都可称为形式化方法。

非形式化方法使用自然语言来描述概念、判断、推理。此方法在证明上的严密性最低。半形式化方法也使用自然语言来描述概念、判断、推理，但也使用了类似形式化方法的手段来增强严密性的证明。

要实现安全保障需求是要付出许多代价的。在任何硬件和软件系统中，导致安全隐患的安全漏洞是很常见的。比如一些操作系统或应用程序被应用到不适当的环境中，或者本身存在严重漏洞，其安全性就会下降。

信息系统安全问题有以下几种来源：

- 需求定义的遗漏和错误；
- 系统设计的缺陷；
- 硬件缺陷，如接线和芯片的缺陷；
- 软件执行错误，如程序错误和编译错误；
- 系统使用或操作中的错误，以及不经意的失误；
- 故意滥用系统；
- 系统硬件、通信部件或其他设备故障；
- 环境影响，包括自然因素和非自然因素；
- 系统升级、维护错误以及停止运转。

安全保障技术可以解决上述来源所引发的安全问题。设计中的安全保障技术应用到需求分析中可以解决第 1、2 和 6 条引发的安全问题。如果安全需求的定义有误，则系统安全的定义肯定也是错误的，那么系统就没有安全性可言了。正确分析系统所面临的安全威胁，在设计中

应用安全保障技术可以发现设计中存在的安全漏洞，以便在系统实现和实施之前纠正错误。

实现中的安全保障技术可处理硬件和软件实现中的错误(第 3、4 和 7 条)、系统维护和升级的错误(第 9 条)、系统滥用(第 6 条)和环境引发的问题(第 8 条)。系统操作过程中的安全保障技术可处理系统使用过程中的错误(第 5 条)以及系统滥用的问题(第 6 条)。

安全保障的目标是表明系统从实现到运行的整个生命周期中是满足定义的安全需求的。因为系统开发的不同阶段使用了不同的安全保障技术，于是可将安全保障技术分为策略的安全保障、设计的安全保障、实现的安全保障和运行的安全保障。

策略的安全保障需要对需求进行严格分析，要证明安全需求的完整性和一致性。首先要标识出系统的安全目标，然后说明安全需求能够应对系统的威胁。当正确的安全需求被定义好，并经过证明和核准后，系统的设计和实现工作才可以有把握地展开。

设计的安全保障指使用安全工程的一些技术方法来合理进行设计，从而实现安全需求。同时还包括如何衡量设计满足安全需求的程度。

实现的安全保障指使用安全工程的一些技术方法在开发和系统运行阶段正确地实现设计，同时还包括如何衡量实现与安全需求一致性程度。

运行的安全保障是通读系统的使用说明，以保证系统不会因为偶然的设置错误而处于不安全状态。

开发人员完成满足安全需求的系统设计，同时提供安全保障的证据以证明设计的确是满足安全需求的，然后就是保证正确地实现了系统的设计。每个设计和实现的修正过程之后就是一个安全保障证明的过程，证明在系统实施的过程中，需求在连续的层次中仍然是满足的。整个过程是迭代的，当安全保障过程中出现问题时，要重新检查受到影响的步骤。

### 2. 建造安全可信的系统

为某种应用而考虑开发系统时，系统就开始了其生命周期。首先简单介绍生命周期。生命周期包括了一系列阶段，每个阶段包括若干工作以及如何管理这些工作。系统设计和编码就是工作的例子，计划、配置以及规范的选择和使用都是管理工作的例子。所有这些工作都贯穿于系统从基本的构想开始到项目的确定、系统开发、系统实施、系统维护、最后到系统退役的整个过程中。通常将生命周期划分为若干阶段，有些阶段与上一个阶段是相关的，而有些阶段是独立的。每个阶段描述本阶段的工作并控制和其他阶段的交互。在项目进行的过程中，理想情况是系统从生命周期的一个状态不断转移到下一个状态，但在实践中，经常有迭代的情况，比如当后面阶段中发现了前面阶段存在的错误或者遗漏时，就需要重新进行前一阶段的工作。

生命周期的瀑布模型是分阶段开发的模型，在开发过程中一个阶段总是在前一个阶段结束以后才开始。瀑布模型包括 5 个阶段：

1) 需求定义和分析阶段。这个阶段将高层次需求更详细地展开，同时在对系统进行整体架构设计的时候也有可能会产生新的具体要求。所以需求定义和系统整体设计未完成之前，两者之间很有可能有一个反复迭代的过程。需求可分为功能需求和非功能需求。功能需求描述系统与运行环境之间的交互。非功能需求是对系统本身的一些限制，会影响设计和实现。需求只是描述“要什么”而不是“怎么做”。

2) 系统和软件设计阶段。进行系统和软件设计时，一般需要建立外部功能规范和内部功

能规范。外部功能规范描述系统的外部特征，如输入、输出和函数的约束；内部功能规范描述使用的算法、数据结构以及需要的内部函数。这个阶段也分为两个子阶段：系统设计和程序设计。系统设计阶段设计整个系统，程序设计阶段设计单个程序。

3) 实现和单元测试阶段。系统实现是在前面系统设计的基础之上实现系统程序，单元测试是测试程序中的单元是否满足其设计规范。

4) 整合和系统测试阶段。系统整合是将经过单元测试的程序组装成完整系统的过程。系统测试是测试整个系统满足系统需求的过程。系统测试也是一个迭代过程，因为在测试中经常会发现问题，然后要进行问题的修正。修改后的程序进行重新组装，然后进行系统测试。

5) 系统运行和维护阶段。系统开发完毕后，投入运行。系统维护包括修正系统在运行过程中发现的错误以及以前发现的尚未修正的错误。

在实际工程项目中，各个阶段之间都会有迭代，因为后一个阶段经常可以发现前一个阶段中存在的不足之处，需要重新进行前一个阶段的工作。将安全保障贯穿于系统开发的整个生命周期中，有助于让系统实现安全需求。使用生命周期模型并不能保障没有错误发生，但有助于减少错误发生的次数。所以，为系统引入安全机制可以提高系统的可信度。建造安全可信的系统，要求对系统设计和实现过程中的每一步都适度地考虑安全保障。

### 3. 需求定义和分析中的安全保障

安全威胁指破坏保密性、完整性或者造成拒绝服务。安全威胁可能来自系统外部，也可能来自系统内部；可能来自授权的用户，也有可能来自非授权的用户。非授权用户可以伪装成合法的用户，或使用欺骗手段来绕开安全机制。安全威胁还可能来自人为错误，或者是不可预测的因素。

每种被识别出来的安全威胁都应该有相应的应对手段。例如，设定一个安全目标，在访问任何系统资源之前，所有用户都必须通过用户标识和身份验证，以此来应对非法使用系统的威胁。有些情况下，安全目标并不足以应对所有安全威胁，这时需要对系统的运行环境做出一些系统假设，比如增加物理保护手段等，以应对所有威胁。将安全威胁映射成安全目标和系统假设，可以部分解决系统安全需求的完整性问题。

安全策略就是一系列安全需求的规范说明，是提供安全服务的一套准则，概括地说，一种安全策略要表明当系统在进行一般操作时，什么是安全范围允许的，什么是不允许的。要准确描述需求并不是一件容易的事情。定义安全策略和安全需求的方法有很多，一种可行的方法就是从现有安全标准中精选出一些可行的需求；另一种方法是结合现有的安全策略和对系统安全威胁的分析得出新的安全策略；第三种方法就是将系统映射到现有的一个模型上。当完成了安全策略的定义和规范，就必须对安全策略的完整性和一致性进行验证。

### 4. 系统和软件设计中的安全保障

设计的安全保障是确认系统设计满足系统安全需求的过程。设计保障技术需要用到需求规范和系统设计规范，是一个检查设计是否满足需求的过程。

模块化和分层的设计和实现方法可以简化系统的设计和实现，从而使系统的安全分析更加可行。如果一个复杂系统有很好的模块化结构，则它的安全分析也将更可行。分层方法也简化了设计，便于更深入地理解系统。另外，撰写设计文档和规范也是必要的，为了进行安

全分析，设计文档中至少应该包括如下三方面的内容：安全函数，外部接口，内部设计。规范可以是非形式化的、半形式化的或者是形式化的。非形式化的规范使用自然语言来描述，半形式化方法也使用自然语言来描述规范，同时使用一个整体方法强加某种限制。形式化方法使用数学语言和可用机器解释的语言。形式化方法的语义可以帮助检查出规范撰写中被忽略的一些问题。

描述规范的方法决定了验证规范所能使用的技术。非形式化和半形式化的规范描述是不能用形式化的验证方法来分析验证的，因为这些规范描述使用的语言不是十分准确。不过还是可以做一些非形式化的验证工作。对于非形式化的规范描述可以验证其是否满足需求，可以验证不同层次的规范文档是否一致。常用的非形式化验证方法有需求跟踪、非形式化对照和非形式化讨论。

需求跟踪是标识在某个规范中的不同部分满足特定安全需求的过程。非形式化对照的作用是展示设计规范与相邻层次设计规范的一致性。将这两种方法结合起来，可以更大程度地保证规范文档完整地、一致地满足为系统定义的安全需求。撰写形式化的设计规范开销很大，所以，撰写形式化设计规范的开发者都喜欢使用一些自动化工具来完成这个任务，比如使用基于证明的技术或者模型检验器。对形式化规范进行需求跟踪可以检查规范描述是否满足需求。在使用形式化方法之前先做非形式化论证有利于为形式化的证明提供思路。

形式化证明技术是一种通用技术，通常基于一些逻辑演算，如谓词演算。这些技术通常是交互式的，有时称为“证明检验者”，这是因为这种技术只是验证证明的步骤是否正确。形式化证明技术被用于证明一个规范满足某个性质，自动化的证明工具可以自动处理规范和相应的性质。

模型检验则是检验特定规范是否满足特定模型的约束。模型检验器是一种自动化工具，对于一个特定的安全模型，模型检验器用于检查一个规范是否满足该模型的约束条件。这种检验常应用于操作系统。

### 5. 系统实现和整合中的安全保障

证明一个实现是否满足安全需求的最好方法就是测试。安全测试的方法可使系统实现和整合过程能有更多的安全保障。

系统是模块化的，在可能的情况下，尽量将与安全无关的功能从实现安全功能的模块中去掉。系统所使用的语言也会对安全保障产生一定的影响。有些语言对安全的实现有很好的支持，使用这样的语言可以避免一些常见的缺陷，且系统更可靠。例如使用 C 语言实现的系统，可靠性有限，因为 C 语言没有适当地限制指针的使用，并且只有最基本的错误处理机制。而支持安全实现的语言能够检查出许多实现上的错误，使用强类型、具有越界检查的、模块化的、具有分段和分段保护的、具有垃圾回收和错误处理机制的编程语言所实现的系统是更可信的，更有安全保障。例如 Java 就是以实现安全代码为目标的程序设计语言。但是有时候使用高级语言效率比较低，此时编程规范可以弥补语言在安全方面的不足。比如限制低级的编程语言只能在不适合使用高级语言的地方使用。

对模块化的系统进行整合时，良好的模块设计和模块接口的设计显得尤为重要，使用一些管理方面的支持工具也将很有帮助。配置管理是在系统开发和使用期间对任何系统硬件、软件、固件、文档和测试文档的变动所实施的管理。一般由若干工具或者手工处理过程组成，

必须执行以下操作：版本控制和跟踪，修改授权，合并程序，实现系统。

有两种典型的测试技术：功能测试和结构化测试。功能测试，也称为黑盒测试，用于测试一个实体满足设计规范的程度。系统测试是对整合后的系统进行的功能测试。结构化测试，也称为白盒测试，其测试用例都是在对代码的分析的基础上得出的。单元测试是程序员在系统整合之前对代码模块进行的测试，一般都是结构化测试。第三方测试，也称为独立测试，是由开发团队之外的其他方进行的测试。

安全测试是解决产品安全问题的测试。安全测试包括三个部分：安全功能测试，主要测试相关文档中描述的安全功能；安全结构测试，主要对实现安全功能的代码进行结构化测试；安全需求测试，主要针对用户需求中的安全需求部分进行测试。一般而言，安全功能测试和安全需求测试是单元测试和系统测试中的一个部分。第三方测试可能会包括安全功能测试或者只包括安全需求测试。安全结构测试可以是单元测试和系统测试的一部分。

#### 6. 系统运行和维护中的安全保障

系统实施完成后进入运行阶段，运行时可能会出现错误，所以还需要对系统进行维护。热修复只是即时修改错误，然后将修正版本发布。常规修复解决不是十分严重的错误，一般是累积到一定程度才发行出去。

### 3.4.2　审计

审计是事后认定违反安全规则行为的分析技术。在检测违反安全规则、准确发现系统发生的事件以及对事件发生的事后分析方面，审计都发挥着巨大作用。

计算机系统审计技术的发展，来源于对访问的跟踪，这些访问包括对保存在信息系统中的敏感及重要信息的访问和对信息系统资源的访问。已经有一些简单工具用来分析审计记录，并检查对于系统及文件的未授权访问。这些工作的前提是日志机制，在日志中记录许多额外信息。

日志就是记录的事件或统计数据，这些事件或者统计数据能够提供关于系统使用及性能方面的信息。审计就是对日志记录的分析并以清晰、易于理解的方式表述系统信息。

日志提供了一种安全机制，它能分析系统的安全状态，能判断某个请求的行为是否会使系统处于不安全状态，或判断一个事件序列是否会导致系统处于不安全(或危及安全)状态。例如，日志记录了所有引起状态转变的事件，系统就能在任何时候重建系统状态。或者只要记录了这些信息的一个子集，就能够消除引起安全问题的某些可能因素，也能为进一步分析提供有用的基础。

审计机制必须记录权限的使用情况，有了这些记录和分析，审计机制能够阻止攻击，因此能确保检测到任何对安全策略的破坏。那么，日志应该记录哪些信息？审计哪些信息？这是两个相互联系的问题。要决定哪些事件和行为应该被审计，就需要知道系统安全方案相关的知识，知道哪些尝试入侵是安全方案所允许的，知道如何检测这些尝试。于是必须知道日志需要记录哪些信息，入侵者一定会使用哪些命令来尝试入侵，入侵者一定会产生哪些系统呼叫，他们会用何种顺序发出这些命令和系统呼叫等。所有事件的日志提供了这些信息，问题是如何辨别信息的哪些部分是相关的，确定需要审计的关键问题。

一个审计系统包含三个部分：日志记录器、分析器和通告器，它们分别用于收集数据、

分析数据和通报结果。

审计机制的分析建立在日志系统所记录的数据的基础上，并且这个机制分析关于系统安全状态的相关信息，决定是否发生了特定系统行为或者是否进入某种特定状态。

审计的目标决定日志记录哪些信息。一般地，审计员希望检测到违反安全策略的行为。假设 Ai 是一个系统中可能行为的集合，安全策略提供了约束 Pi。为保证系统安全，设计必须满足约束 Pi，也就是说必须审计那些使约束失败的功能调用。

例如 Bell-LaPadula 策略模型将安全等级 Li 线性排列。主体 s 有等级 L(s)，客体 o 有等级 L(o)。在这个策略下，当 L(s)< L(o)时，s 读取 o，或者当 L(s)> L(o)时，s 写 o 是非法的。相应的约束就是：s 读 o 推出 L(s)≥L(o)，s 写 o 推出 L(s)≥L(o)。

违反安全策略的审计仅仅要求审计从一个主体到一个低级客体的写，或者从一个高级客体的读，并检查违反这些约束的行为。日志必须包括涉及主体和客体的安全等级、行为和结果。从这些日志记录可以非常容易地测试是否有违反约束的行为。这里并不需要记录客体和主体的名字。但在实际情况中，本地安全策略常要求安全分析员标识出违反约束的客体和试图违反约束的用户。如果没有静态原则，主体可以把它控制的任何主体、客体的安全级别或分类改变成不比它自己高的级别。因此需要记录操作命令、新旧安全等级和新旧分类。

从这个例子可以得出，基于 Bell-LaPadula 系统的审计需要记录以下内容：对于读和写，需要记录主体的安全等级、客体的安全等级和操作的结果；对于无静态原则的系统，需要记录主体或客体，及其新旧安全等级、改变安全等级的主体的安全等级和改变结果。

如果审计结果表明有违反安全规则的行为，则系统是不安全的。但是，如果审计结果表明没有违反安全规则的行为，则系统仍可能是不安全的。因为，如果系统的初始状态是不安全的，结果将是(或者很可能是)不安全的。因此，如果用审计来检测系统是否安全，而不是检测攻击入侵的操作，就还需要获取系统的初始状态，需要一开始就将系统状态转换中会改变的信息记录到日志。

有一个关键的问题是如何来操作日志。比如，哪种数据应该放入日志文件中，数据该如何表述。但是现实中，许多系统日志数据模糊不清使得不能清晰地说明日志记录所记录的内容，所以使用基于语法的方式来定义日志内容。比如使用 BNF 的表达方式，使设计者必须定义日志内容的语法和语义，这样分析员就能使用语法来分析日志记录。

当检测出所有可能的违反安全规则行为时，并不代表能够设计出有效的审计系统。许多违反安全的行为是在已有系统上出现的，这些系统在设计时并没有考虑安全因素。此时，审计可能有两个不同目标，第一个目标是检测对某个安全策略的任何攻击，此目标着重安全策略。基本思路是判定某个状态是否违反了安全策略。审计机制必须结合到已存在的系统中。分析员必须分析系统，判定什么操作和设置与安全策略一致。然后设计某种机制来检查操作和设置是否真正与策略一致。第二个目标是检测已知的企图违反安全规则的操作，此目标注重特殊行为。许多情况下，安全策略并没有明确地定义，然而，某些行为明显是不安全的，如洪泛攻击或未授权的用户访问计算机系统，会违反潜在的安全策略。此时，分析员可通过命令的特定次序或系统状态的特征来寻找并发现针对安全的攻击。

不同系统采用不同方式处理日志。许多系统默认情况下是记录所有事件。对于不需要记录的特定事件，系统管理员要进行显式指定，这就会带来日志膨胀问题。

对于设计时考虑安全的系统，将把审计机制结合到系统的设计和实现中。典型的做法是

提供某种语言或界面，以允许系统管理员配置系统来报告特定事件，监控特定主题或监控对特定客体的访问。

对于设计时没有考虑安全的系统，其审计系统能用于检查出严重的安全攻击行为，但一般不记录事件的某些细节或事件类型，以使安全管理员难以判定是否有违反安全规则的行为。

### 3.4.3　系统评估

系统评估是一种以具体的安全功能要求和安全保障证据为基础，对系统进行可信度测量的技术。通过评估，可指出系统满足具体标准的程度。评估时所采用的标准依赖于评估的目标以及所采用的评估方法。可信计算机系统评估标准(TCSEC, Trusted Computer System Evaluation Criteria)是第一个被广泛使用的形式化评估方法，随后的评估方法都建立在 TCSEC 的基础之上，或是对 TCSEC 的改进。

TCSEC 标准是计算机系统安全评估的第一个正式标准，具有划时代的意义。该准则于 1970 年由美国国防科学委员会提出，并于 1985 年 12 月由美国国防部公布。TCSEC 最初只是军用标准，后来扩展至民用领域。TCSEC 是按照评估等级来组织的，在各个评估等级的描述中定义了功能需求和安全保障需求。

TCSEC 将计算机系统的安全划分为 4 个等级、7 个级别。每个等级都包含了相应的功能需求和安全保障需求。随着评估级别的提高，功能需求和安全保障需求都会不断地增加和提高。

D 类安全等级：D 类安全等级只包括 D1 一个级别。D1 的安全等级最低，只为文件和用户提供安全保护。D1 系统最普通的形式是本地操作系统，或者是一个完全没有保护的网络。

C 类安全等级：C 类安全等级能够提供审慎的保护，并为用户的行动和责任提供审计能力。C 类安全等级可划分为 C1 和 C2 两类。C1 称为自主保护。C1 系统的可信任运算基础体制(TCB, Trusted Computing Base)通过将用户和数据分开来达到安全的目的。在 C1 系统中，所有用户以同样的灵敏度来处理数据，即用户认为 C1 系统中的所有文档都具有相同的机密性。C2 称为受控访问保护。C2 系统比 C1 系统加强了可调的审慎控制。在连接到网络上时，C2 系统的用户分别对各自的行为负责。C2 系统通过登录过程、安全事件和资源隔离来增强这种控制。C2 系统具有 C1 系统中所有的安全性特征。

B 类安全等级：B 类安全等级可分为 B1、B2 和 B3 三类。B 类系统具有强制性保护功能。强制性保护意味着如果用户没有与安全等级相连，系统就不会让用户存取对象。B1 称为标签安全保护。B1 系统满足下列要求：系统对网络控制下的每个对象都进行灵敏度标记；系统使用灵敏度标记作为所有强迫访问控制的基础；系统在把导入的、非标记的对象放入系统前标记它们；灵敏度标记必须准确表示其所联系的对象的安全级别；当系统管理员创建系统或者增加新的通信通道或 I/O 设备时，管理员必须指定每个通信通道和 I/O 设备是单级还是多级，并且管理员只能手工改变指定；单级设备并不保持传输信息的灵敏度级别；所有直接面向用户位置的输出(无论是虚拟的还是物理的)都必须产生标记来指示关于输出对象的灵敏度；系统必须使用用户的口令或证明来决定用户的安全访问级别；系统必须通过审计来记录未授权访问的企图。

B2 称为结构化保护。B2 系统必须满足 B1 系统的所有要求。另外，B2 系统的管理员必须使用一个明确的、文档化的安全策略模式作为系统的可信任运算基础体制。B2 系统必须满

足下列要求：系统必须立即通知系统中的每一个用户所有与其相关的网络连接的改变；只有用户能在可信任通信路径中进行初始化通信；可信任运算基础体制能够支持独立的操作者和管理员。

B3 称为安全域。B3 系统必须符合 B2 系统的所有安全需求。B3 系统具有很强的监视委托管理访问能力和抗干扰能力。B3 系统必须设有安全管理员。B3 系统应满足以下要求：除了控制对个别对象的访问外，B3 必须产生一个可读的安全列表；每个被命名的对象提供对该对象没有访问权的用户列表说明；B3 系统在进行任何操作前，要求用户进行身份验证；B3 系统验证每个用户，还会发送一个取消访问的审计跟踪消息；设计者必须正确区分可信任的通信路径和其他路径；可信任的通信基础体制为每个被命名的对象建立安全审计跟踪；可信任的运算基础体制支持独立的安全管理。

A 类安全等级：A 系统的安全级别最高。目前，A 类安全等级只包含 A1 一个安全类别，称为验证保护。A1 类与 B3 类相似，对系统的结构和策略不做特别要求。A1 系统的显著特征是，系统的设计者必须按一个正式的设计规范来分析系统。对系统分析后，设计者必须运用核对技术来确保系统符合设计规范。A1 系统必须满足下列要求：系统管理员必须从开发者那里接收到一个安全策略的正式模型；所有安装操作都必须由系统管理员进行；系统管理员执行的每一步安装操作都必须有正式文档。

#### 1. TCSEC 的安全功能需求

身份识别和认证需求详细说明了系统是如何标识用户的身份，以及认证该用户的身份。同时还处理认证数据的粒度(每一组或每一个用户等)、认证数据的保护以及与审计相关的身份标识问题。

自主访问控制需求确定了一种访问控制机制，访问控制的权限由访问对象的拥有者来自主决定，也可由被授权控制对象访问的人来决定。拥有者能够决定谁应该拥有对其对象的访问权及内容。自主访问控制的实现可以基于主体或者客体来进行。

强制访问控制需求在低于 B1 级的评估等级中并不要求，个人用户不能改变这种控制。强制访问控制往往根据一个具体的安全模型，通过预设的规则来进行。标记是强制访问控制的基础，标记可以与主体和客体绑定在一起的方式存在，也可以存放在数据表中，使用时根据主体和客体的特征字段进行查找。

审计需求要求系统中存在一种审计机制并保护审计数据。该需求定义了审计记录必须包含的内容以及审计机制必须记录的事件。随着评估级别的提高，审计需求不断扩展。

TCSEC 还提出其他一些需求，如客体重用需求和可信路径需求。客体重用需求主要是为了防止攻击者从可重用客体中收集信息，比如从内存和磁盘中获取信息。该需求要求在释放一个可重用客体时，要撤消前一个用户对该客体的访问权限，并阻止新用户读取前一个用户留在该客体中的信息。可信路径需求提供一条保证在用户和 TCB 之间通信的路径。

#### 2. TCSEC 的安全保障需求

配置管理需求要求验证所有的配置项，验证所有的文档和代码之间的一致性，以及验证用于生成 TCB 的工具。该需求只对等于或者高于 B2 的评估等级进行要求。

可信软件发布需求要求保证软件原版和在线版本之间的一致性和完整性，并保证用户的

接收过程是安全的。该需求只针对 A1 评估等级。

系统体系结构需求要求系统模块化、复杂性最小化等。目标是使 TCB 尽可能简短。该需求只对 C1 和 B3 之间的评估等级进行要求。

设计规范和验证需求在不同的评估等级中差别很大。评估等级 C1 和 C2 没有此要求。B1 需要一个非形式化的、与相关公理一致的安全策略模型；B2 需要一个形式化的安全模型，该模型与相关公理的一致性必须是可证明的，并且系统具有一个描述性的高层规范 DTLS(Datagram Transport Layer Security)；B3 进一步要求说明 DTLS 和安全策略模型之间的一致性。

测试需求要求测试结果和目标一致，系统能抵御攻击并在修正系统漏洞后再重新测试。在高等级评估中，还要求采用形式化方法搜索通道。

产品文档需求在所有的评估等级中都要求存在。该需求分为安全特征用户指南和可信设施手册。安全特征用户指南需求要求描述保护机制，各种机制之间如何交互，以及如何使用这些保护机制；可信设施手册需求要求描述产品安全运行的需求，包括生成、启动和其他规程。

#### 3. TCSEC 的评估过程

评估分为三个阶段：申请、初步技术审查 PTR、评估。评估由政府资助的评估单位进行。如果政府不需要一个产品，该产品的评估申请将被拒绝。否则进行下一步 PTR：详细讨论评估过程、评估进度、开发过程、产品技术内容和需求等，然后决定何时派遣评估工作组以及具体的评估进度表。这个过程实际上是为评估做的准备工作。到评估阶段，又可以细分为三个小阶段：设计分析、测试分析、最终评论。在每个小阶段的评估结果都必须提交技术审查委员会 TRB 进行审查，只有评估的结果得到认可才能进行下一个阶段的评估。

## 3.5 网络安全

计算机通过互联网进行信息的交换，这形成了很多交互操作。网络上的用户怎样访问本地的系统，本地的系统用户怎样访问网络以及怎样保护本地的数据在网络中传输。因此，网络安全不仅是应用密码学，还包括网络级访问控制和入侵检测目标。

本节将介绍安全问题对网络的挑战，网络安全对计算机安全的影响和依赖。以基本的 Internet 安全协议 IPsec 和 SSL/TLS 为例，进行网络安全协议的设计。还讨论防火墙技术和入侵检测系统的原理和局限等。

### 3.5.1 网络威胁模型

计算机网络是分布式系统中节点之间传输数据的通信基础设施。一个节点上的应用程序将要发送的数据事先准备好，用一个电信号或光信号序列传送出去，接收端进行重组，并发给应用程序。网络协议必须找到一条从发送者到接收者的路径，必须处理数据的丢失和损坏以及失去的连接，例如施工人员切断电话线缆。成熟的工程实践方法是逐一解决这些问题，并使用分层体系结构，把应用协议放在顶部，把有关物理传输的编码信息放到底部。

ISO/OSI 安全结构(国际标准化组织，1989)定义安全服务(Security Service)来阻止网络安

全威胁。安全服务由安全机制(Security Mechanism)来执行，提供的这些服务的机制大部分来自密码学。典型的例子有加密、数字签名、完整性检查功能等。密码保护有一个特性：当 N 层安全协议在其下层不安全的协议的上面运行时，N 层的安全协议不会受到危害。对这种特性存在例外。当你的目标是匿名，而且你采取了预防措施来隐藏在某一层的参与者的身份时，则由较低层的协议添加的一些数据仍可能泄露关于消息的来源和目的地的信息。

并非所有安全问题都能用密码学解决，进一步的防御是访问控制机制(防火墙)和入侵检测系统。设计协议对付拒绝服务攻击或限制这种攻击不放大是网络安全中的另一个需要关注的问题。

#### 1. 威胁模型

网络上的攻击者可以是被动的或主动的，被动攻击者仅监听通信。当攻击者仅监听通信时，我们讨论偷听、搭线窃听(wiretapping)或嗅探(sniffing)。当攻击者无法读取私人消息时，通信量分析可试图确定通信参数。攻击者会试图识别相同的源(链接)或者找出通信双方以及通信频率。在移动服务中，攻击者也许还对用户的位置感兴趣。

主动攻击者也许会修改消息、插入新消息或者毁坏网络管理信息，如在域名系统(DNS)中的名字和 IP 地址。在欺骗攻击中伪造消息发送者的地址。在洪泛攻击中，大量的消息被指向受害者。在蹲守攻击(Squatting Attack)中，攻击者要求定位受害者。主动攻击并非比被动攻击更难实施。事实上，伪造发送人邮件比拦截别人的邮件更加容易。

攻击者会试图了解你的网络内部结构，并使用这种信息来发动攻击。来自于网络管理协议的信息安全非常敏感，因为这些协议收集的是有关节点的负荷以及可用性的诊断信息。同时，这些协议需要高效的使用网络。如果过分保护这些协议，网络提供的服务质量会大大降低。

#### 2. 通信模型

安全协议在抽象层中描述。消息直接在主角之间传递，我们没有考虑这种交换的精确性。安全协议的分析常常在这种模型中进行。Internet 用云朵形状表示。信息可被任何决定要发送或修改的人发送或修改。攻击者可通过协议分析在正常的网络模型中控制所有的信息流动。

这样一个抽象模型并非一个最好的能解决所有安全问题的模型。在安全分析中，可假定有一个潜在真实对手。我们假定对手不能控制整个网络，只能读那些直接发向他的地址的信息。但他可以任意修改发送者的地址。在设计协议时，我们可能发现网络中的实体(比如防火墙或 NAT)会妨碍通信者之间运行他们的协议，需要特别予以注意。

#### 3. TCP 会话劫持

与客户机服务器 B 创建一个 TCP 会话，客户机 A 需要先启动下面三步握手协议。

1) A->B：SYN，ISSa；

2) B->A：SYN|ACK，ISSb，ACK(ISSa)；

3) A->B：ACK，ACK(ISSb)。

SYN 和 ACK 表明各自的位已经被设置好。ISSa 和 ISSb 是 32 位序列号。它们的关系是

ACK(ISSa)=ISSA+1 和 ACK(ISSb)=ISSb+1。

假设消息总是发送到指定的接收者，而且在传输过程中不能被观察到。攻击者只能插入假的源地址和他自己的消息。只要序列号是随机的，这个协议就是安全的。攻击者就需要猜测发送到另一方的序列号。因此，RFC 793 指定每 4 微秒让 32 位的计数器低位增加 1。但 Berkeley 导出 Unix 内核每秒增加 128，每个新连接增加 64。因此，就没有太多的随机性来混淆攻击者。

早在 1985 年就有利用这种执行决策的攻击，并在 1989 年被推广。攻击者 C 先和他的目标 B 开始一个真实的连接，同时收到一个序列号 ISSb。然后，攻击者假扮 A 发送一个在源地址填入了 A 的地址的包：C(A)->B：SYN，ISSc。

B 回应：B->A：SYN|ACK，ISSb'，ACK(ISSc)到合法的 A。C 不能看到这条消息，但他使用 ISSb 来预测当前 ISSb'，发送 C(A)->B：ACK，ACK(ISSb')。

如果猜测正确，B 将和 A 连接，尽管是 C 在发包。C 看不到这个会话的输出，但它也许可使用 A 在 B 上的权限执行命令。这种攻击方法可以在 Unix 环境下运行，在此环境下攻击者可从可信主机 A 处骗取消息。类似 RSH 使用基于地址验证的协议，假设从一个可信主机登录的用户被验证，这样会存在风险。

让防火墙封锁来自局部来源地址的所有 TCP 包，防御这种攻击。这种方式只能在所有你信任的主机都位于一个局域网中时才起作用。如果还有在 Internet 中受信任的主机，防火墙就不得不阻塞所有使用 TCP 的协议和基于地址的认证。更佳的解决方案是完全避开基于地址的认证，采用密码认证。

#### 4. TCP-SYN 洪泛攻击

在应答了第一个 SYN 包后，B 服务器会存储 ISSb 序列号，来验证客户端发的 ACK 攻击。攻击者在使用 TCP SYN 洪泛攻击时，初始化大量的 TCP 请求(SYN 包)，但并不完成协议的运行，直到 B 服务器达到了它的半开连接的极限，至应答新的请求限制为止。将修改 TCP 握手协议以便允许服务器动态丢弃保留的状态。作为 TCP 会话劫持攻击的一部分，C 能够发动 SYN 洪泛攻击 A，因此 A 不能处理来自 B 的 SYN-ACK 包，也不能丢掉攻击者试图打开的连接。

### 3.5.2　网络协议安全

#### 1. 协议涉及原则

ISO/OSI 结构的 7 层模型是分层网络协议的常见框架，从底层至顶层分别是物理层、链路层、网络层、传输层、会话层、表示层和应用层。分层模型为网络安全的研究提供了实用的抽象概念。顶部的安全服务可用来满足特定应用的要求。不同的应用都需要自身的安全协议。底部的安全服务可用来保护所有的高层通信，减轻应用层协议设计者安全方面的设计。然而，一些应用可能发现这些保护措施不能很好地满足它们的要求。

在分层模型中，N 层的对等实体(Peer Entity)使用 N 层协议。N+1 层协议看到的是在 N 层上的一个虚拟连接，并且不会考虑任何底层的情况。N 层协议建立在低层协议之上。N 层协议的消息成为 N 层协议数据单元(PDU，ProtocolData Unit)。N 层协议数据单元可能被分段

或者经过其他处理。对结果增加头部和尾部使其成为N-1层协议数据单元((N-l)-PDU)。这些N-1层协议数据单元的接收者使用报头和报尾中的信息来重组N层协议数据单元((N)-PDU)。

有两个重要的选择来实施N层协议调用的N-1层的安全服务。较高层的协议能够察觉更低层的安全协议，或者安全协议是透明的。第一种情况下，高层的协议不得不改变它的调用来访问提供的安全设施。第二种情况下，高层协议不必改变。在两种情况下，N-1层协议数据单元的帧头是一个比较方便存储安全相关数据的地方。

网络协议栈有4层，从下层至上层依次是链接层、互联层、传输层和应用层。应用层的协议有telnet、ftp、http、smtp或者SET(Secure Electronic Transaction)这些协议。在传输层有TCP(Transmission Control Protocol，传输控制协议)和UDP(User Datagram Protocol，用户数据报协议)协议。网络层有IP协议。TCP和UDP使用端口号来标识协议数据单元所属的主体。常用的端口号是2l(ftp)、23(telnet)、25(smtp，发送邮件)、110(pop3，接收邮件)、143(imap，接收邮件)、80(http)、443(https，安全网页)和53(DNS，名称查找)。链接层协议使用特殊网络技术。

TCP和IP以及UDP和管理协议ICMP是网络核心。最初，这些协议可保证友好和合作的用户能通过可靠的网络进行连接，因此完全不考虑安全性。如今，TCP/IP被广泛地使用，并要求更强大的安全性。因特网工程任务组(IETF, Internet Engineering Task Force)已经对因特网安全协议和请求注释(RFC，Requests for Comments)进行标准化。IETF对大量使用中的安全协议进行修订，并发布新的网络安全协议。

2. IP 安全

IP是一个无连接无状态的传输IP包的协议，这些IP包就是网络层的协议数据单元，核心IP规范提供尽力的服务。IP是无连接无状态的，因此每个数据报被视为一个不依赖于其他任何IP数据块的独立实体。无法保证包能投递到目的地，没有机制来维持包的顺序，也没有安全协议。IPv4作为RFC 791在1981年发布。之后，互联网不断发展，结果IP协议不得不适应新要求。IPv6在RFC1883中被明确提出。当讨论IP安全机制的时候，将参考IPv6。

IP的安全结构(IPsec)在RFC2401中有介绍。IPsec在IPv4中是可选的，但在IPv6中却是强制要求的。IPsec包含两个主要的安全机制，在RFC 2402中描述的IP认证报头(AH, Authentication Header)和在RFC 2406中讲述的IP封装安全有效载荷(ESP, Encapsulating Security Payload)。IP安全架构不包括阻止通信量分析的机制。

IP认证报头只保护IP包的完整性和可靠性，但不保护机密性。最初是由于政治原因推荐使用它。20世纪90年代，加密算法的输出限制为只认证机制创建机会。现在这些输出限制基本上被消除，因而推荐只使用ESP来简化IPsec的执行。

ESP提供机密性、数据源认证、数据完整性以及一些重放保护。ESP包由以下几个部分组成。

1) 安全参数索引(SPI，Security Parameters Index)是一个32位的域，唯一用于标识数据块与目标IP地址和安全协议的安全结合体。

2) 序列号是个无符号32位域，包含一个计数值，这个值必须被发送者计入，而由接收者做判断处理。

3) 有效载荷数据是一个包含传输层PDU的可变长度域。

4) 填充区是一个可选的域，它包括为加密算法填充的数据；数据的长度被填充为算法块的整数倍长。

5) 填充长度。

下一个报头是传输层 PDU 的类型。

认证数据以 32 位字长为单位，长度可变。包含一个完整性校验码(ICV，Integrity Check Value)，由 ESP 包减去认证数据(Authentication Date)计算得到。

SPI 和序列号构成了 ESP 的报头，在有效载荷后的域是 ESP 的报尾。ESP 可使用两种模式。在传输模式中，上层协议框架，例如来自 TCP 或者 UDP 的框架被封装在 ESP 中。而 IP 报头并未加密。传输模式为两个终端主机间交换的包提供端到端的保护。当然，两个节点都必须支持 IPsec。

在隧道模式(tunnel mode)中，完整的 IP 数据报与安全域一起被视为外部 IP 数据报的新的有效载荷。原始内部 IP 被封装到外部 IP 数据报，IP 隧道因此可以描述为 IP 中的 IP。隧道模式可用在 IPsec 已在终端主机的网关上实现的情况下。终端主机不需要支持 IPsec。网关可以是一个边界防火墙或路由器。这种模式提供网关到网关的安全，而非端到端的安全。另一方面，我们使通信流保持了机密性，使得中间路由对于 IP 数据报不可见，并且原始来源和目标地址被隐藏。

通常，破解或验证一个 ESP 包的系统必须知道使用的算法和密钥以及初始化矢量。这种信息都存储在一个安全关联(SA，Security Association)中。安全关联通常是在两个通信主机之间建立的单项逻辑连接，定义了双方如何使用安全服务进行通信。两台主机之间的双向通信需要两个安全关联，每一个 SA 保护一个方向的数据。因此，通常成对创建安全关联。

一个安全关联是由 SPI(AH 和 ESP 报头携带)、IP 目的地址以及安全协议(AH 或 ESP)标识符唯一定义。它还包含像算法标识、密钥、密钥时限和可能的 IV 等相关联的密码数据。还有一个序列号计数器和一个反应答窗口。SA 也表明是工作在隧道模式还是传输模式。活动 SA 表保存在 SA 数据库(SAD)中。SA 还能够合并起来，例如，IPsec 隧道的多层次嵌套。每个隧道可在不同路由的 IPsec 网关开始和结束。

如果节点数目很小，而且没有增加到适当的支持 IP 安全协议的主机网络数量，就可以手工创建 SA。手动加密的备选方案之一就是因特网密钥交换协议。IKEv2 是一个正在制定中的互联网密钥交换协议。它的安全目标就是实体认证和建立新的可以共享的密钥。使用共享密钥来获得进一步的密钥。其安全目标还包括所有密码算法的安全协商，例如，认证方法、密钥交换方法、解密算法和哈希算法。一种 Cookie 机制(不要和 http 的 cookie 混淆)用来提高对拒绝服务攻击的抵抗能力。可通过选项将保密性设置为不完全安全或绝对安全。

IPsec 提供很多选项和参数来灵活地创建密钥。IP 安全协议需要大量的对称密钥。每个 SA 都有一个密钥。IKE 在两个阶段中起作用。阶段一，设置一个安全通道来传输下一阶段的 SA 系统，以及错误信息和传输通信量。这个阶段包含可靠的实体鉴别和密钥交换。第一阶段的协议有两种变体。一种是耗时长但更安全的主模式，一种是更快的质询模式。主模式和质询模式都由多种认证机制构成。基于签名和共享(对称)密钥的认证已在实际中部署。阶段二，通常使用的 SA 开始通信。通过第一阶段建立的安全通道快速通信。第一阶段允许运行许多第二阶段。

IPsec 的安全服务不与任何特殊的密钥管理协议相关。如果发现一种密钥管理协议有缺陷，

这个协议可以替换为其他协议而不会影响 IP 安全协议的工具。

IPsec 策略决定了 IP 数据报的安全处理。支持 IPsec 的主机有一个安全策略数据库(SPD, Security Policy Database)。SPD 会查看每个输入输出的数据报。在 IP 数据报中的这个域和 SPD 条目的域匹配。匹配基于源和目的地址(范围地址)、传输层协议、端口号等。一个匹配识别一个或者一组 SA(或者更新的 SA)。管理和实施 IPsec 策略较为复杂，是网络协议的研究热点。

### 3. 安全套接层和传输层安全协议

TCP 协议提供两个节点之间可靠字节流通信。TCP 是一种有状态的面向连接的协议，它检测丢包时间和包的到达次序颠倒时间，并丢弃重复的数据。TCP 在建立两个节点间的会话时，可实现基于地址的实体认证。但它选择了一种很容易受到攻击的执行方式。TCP 缺乏强有力的密码实体认证、数据完整性和机密性。这些服务是在 SSL 协议中引入的。它最初由 NetScape 公司开发，主要保护 WWW 通信安全。RPC 2246 关于传输层安全协议(TLS，Transport Layer Security Protocol)基本上与 SSL 的第三个版本一样。于是，这个协议就变成了众所周知的 SSL/TLS。RFC 3268 为 TLS 定义了基于 AES 的加密算法。

在 IP 协议栈中，SSL 位于应用层协议和 TCP 协议之间。因此，SSL 能够依赖有 TCP 保证的属性，例如它自己就不必关心数据的可靠传递。像 TCP 一样，SSL 也是有状态的和面向连接的。SSL 会话状态包括执行密码算法所要求的信息，例如会话标识符、密码程序组的说明、共享的密钥、证书、协议使用的随机数等。为减少由密钥管理引起的系统开销，一个 SSL 会话可以包括多个连接。有特色的实例是客户和服务器之间的 http 1.0 会话，其中为传输复合文档的每一个部分都要建立一个新连接。对于每个连接，仅仅是状态信息的一个子集发生变化。

SSL 由 SSL 记录层(Record Layer)和 SSL 握手层(Handshake Layer)两部分组成。SSL 记录层从上层协议获得数据块，然后应用在现行会话状态中的密码说明定义的密码变换。从本质上讲，SSL 记录层提供一种类似于 IPsec 的服务，IPsec 安全关联和 SSL 状态相似。SSL 握手协议建立会话状态的密码参数。

SSL/TLS 包含一个握手协议，客户端和服务器利用它来商定密码组，确立必需的加密资料，进行相互认证。目前 SSL 是使用最广泛的因特网安全协议，所有的主流网页浏览器都支持。SSL 在应用协议和 TCP 之间加上了一个安全层，因此应用程序必须明确地要求安全，应用程序代码必须进行改变。

客户端和服务器必须保护它们创建的安全内容参数(或 IPsec SA)。否则，用 SSL 或 IPsec 提供的安全将受到威胁。密码保护不会受到来自通信网络下方层的威胁，它很可能受到来自网络节点操作系统的下层的威胁。

### 4. 域名系统 DNS

应用程序通过 DNS(Domain Name System)解析域名和 IP 地址的映射关系。为了通信，我们需要符合 DNS 名字的 IP 地址。这种信息由 DNS 名称服务器维护。DNS 查询出 DNS 名称对应的 IP 地址。反响检查就是提供 IP 地址对应的 DNS 名称。攻击者会损坏这些信息来使用户错误地连接到伪造的地址，还使一些地址不能使用，甚至造成更严重的破坏。如果其他服务器用这些损坏的结构来升级它们的缓存，会发生错误。

原始的 DNS 查询协议使用了一种非常简单的方法来阻止欺骗。安全的 DNS 服务正在不断部署。

### 3.5.3　防火墙技术

密码机制用于保护传输数据的机密性和完整性；认证协议验证数据的来源。如果要控制允许哪些通信进入系统(进入过滤)，或者传出系统(外出过滤)，可通过部署防火墙来实现。

防火墙是一种控制网络双方之间的传输流的网络安全设备。防火墙通常安装在整个组织内部网与因特网之间。它们也常安装在内部网内来保护单个部门。例如，一个公司可能会在它的某部门的子网和公司主网之间安装防火墙。

防火墙应该可以控制网络传输和保护网络。所有传输只有通过防火墙，才能进行有效的保护。拨号线和无线局域网是防火墙不能保护的入口点。防火墙可以同意或阻止访问服务，在同意访问前验证用户或机器，以及监视进出网络的传输。防火墙在 20 世纪 90 年代早期开始流行。这时，很多终端不能保护自己的 PC 机。它强制将安全中心转移到网络边界。

防火墙用于保护系统，使外部网络中的第三方不能访问那些只能在内部网中可以得到的信息。它们也能限制从内网到外网服务的访问，这些服务对组织的工作来说是危险的或者不是必需的。防火墙可以决定通过虚拟专用网(VPN, Virtual Private Network)路由敏感的通信。VPN 创建一个在组织子网没有直接连接的网关之间的安全连接。所有在两个子网之间的传输都必须通过那些增加了安全措施的密码保护的网关。防火墙也能作为网络地址翻译器(NAT)，在公共 IP 地址后面用私有地址隐藏内部机器，并为内部服务器将公共地址转换为私有地址。

在每个网络层，我们都能找到用来控制访问的参数。在 OSI 的第 3 层，我们有源 IP 地址和目的地 IP 地址。在 OSI 的第 4 层有 TCP 和 UDP 的端口号。许多防火墙假设端口号决定服务，但这并不一定正确。在 OSI 的第 7 层，有很多与应用程序关联的信息：邮件地址、邮件内容、网页要求、可执行文件、病毒和蠕虫、图片、用户名和密码等。而这只是其中的一小部分。通常，网络边界的机器可用来控制对网络的访问，对离开网络的数据流加密码保护，或者隐藏网络的结构。

#### 1. 包过滤

包过滤在 OSI 的第 3 和第 4 层工作。这个规则具体指明哪些包允许通过防火墙，哪些应该阻止，该规则应用到每个包。典型的规则详细说明了源 IP 地址和目标 IP 地址，以及 TCP 和 UDP 的源/目标端口号。也可以定义双向通信规则。这样的防火墙可以用 TCP/IP 包过滤发送程序来检查每个包的报头来实现通过或者阻止包。

简单的规则可以被强制使用，普通的协议很难对其进行处理。例如，当客户端将 ftp 请求发送到 ftp 服务器时，防火墙不能为了某个请求连接的数据包对服务器进行连接。我们拥有针对来自端口 20 的所有包的通用规则，或者源自特定 IP 地址的所有端口 20 的包的规则，但我们不能动态地定义规则。

#### 2. 状态包过滤器

状态包过滤器能够明白请求并做出回复。例如，它们知道关于 TCP 开放序列的(SYN, SYN-ACK, ACK)模式。规则通常只针对一个方向上的第一个包。在第一个包出去之后创建状态。通信中后续的包就可以自动处理了。状态防火墙支持大量协议的过滤规则，而不仅是简

单的包过滤，例如 ftp、IRC 或者 H323。

### 3. 电路级代理

电路级代理有与包过滤差不多的规则，但是并不路由所有的包。规则决定了允许哪些连接，封锁哪些连接。允许的连接会在防火墙和目标之间创建一个新连接。这种防火墙在这里提及，是由于它很少在实际中使用。它的函数性和状态包过滤器差不多，但效能更低。

### 4. 应用层代理

防火墙可以监控每个应用层协议，代理执行防火墙上协议的服务和客户部分。当一个客户端连接到防火墙时，防火墙的代理就像服务器一样使请求生效。代理是另一个受控调用的例子。

例如，一个邮件代理可以过滤掉病毒、蠕虫和垃圾。如果允许客户端的请求，代理就作为客户端，连接到目标服务器。通过防火墙的回应也由代理处理和检查。代理服务器仅是外部看见的结构。它在传输中使用排除过滤的方法，例如，去掉邮件的附件。

应用层的代理通常在固化 PC 上运行。应用程序代理可在进或者出的通信量中提供关闭控制。在这个方面，应用层代理提供了高的安全级别，它所提供的配置比较合适。在底侧，有大量对每个连接的处理，配置也更困难。在这方面，应用层代理没这么安全，防火墙安全方面的漏洞已经出现。相对于包过滤器来说，它的性能更差，而花费更高。此外，你需要为每个希望保护的服务准备一个代理服务器。

包过滤器的行为可比作通过电话号码挂断电话。它们对特定的号码锁定呼叫。例如，收费号码。而应用层代理就像通过监听谈话的电话监听器。

### 5. 防火墙策略

宽松的策略允许所有通信，但是封锁一些比较危险的服务如 telnet 或 snmp，或者封锁通常用来攻击的端口号。这非常容易犯错。如果你忘记封锁一些必要的资源，这些资源很容易被非法利用。限制性的策略封锁所有的通信，而仅开放需要使用的东西，如 http、pop3、smtp 或 SSH，这是一种更安全的做法。如果你封锁了一些需要的资源，会产生不利影响，然后你就可以允许这些协议。策略通常被看成有肯定和否定表项的访问控制列表。典型的防火墙规则集：允许从内部网络到互联网：http、ftp、SSH、DNS；允许从任何地方到邮件服务器：smtp；允许从邮件服务器到互联网：smtp、DNS；允许从内部到邮件服务器：smtp、pop3；允许回应包；封锁其他所有资源；在实际中部署防火墙，定义和管理规则集是关键问题。

### 6. 边界网络

邮件服务器需要进行外部访问来从外部接收邮件。因此它应该位于防火墙内部。仅防火墙能保护邮件服务器从外部访问的权限。邮件服务器也需要内部权限来从因特网收邮件。因此它应该在防火墙内部。如果试图阻止蠕虫和病毒在网络上传播，或阻止机密文档泄露，作为一种解决方案，你可以创建一个边界网络，也被称作非警戒区(DMZ，Demilitarized Zone)。可用在有服务需要从防火墙的内部和外部都访问的情况。除邮件服务器外，Web 服务器和名称服务器也需要同样的配置。

### 7. 局限性和问题

防火墙不能保护内部不受威胁。封锁服务可能会造成用户的不便，网络诊断将比较困难，很难支持某些协议。包过滤防火墙不能提供任何基于内容的过滤。如果邮件是被允许的，那么包含病毒的邮件也会被允许通过。甚至应用层代理防火墙也不能完全地检查内容。防火墙不知道操作系统和应用程序的漏洞。目前大量服务使用 http 的 80 端口。因此，判断通过这个端口的通信是否是合法非常困难。

协议隧道，即通过另一种协议发送这种协议的数据，将使防火墙失效。由于越来越多谨慎的管理员必须开启 80 端口外，关闭其他几乎所有的端口，越来越多的协议也通过 http 隧道来穿过防火墙。另一个可选用的隧道是 SSH 协议。

不能检查和过滤加密通信，因此很多协议(像 https 和 SSH)都提供端到端的密码保护，这样就不能被防火墙监视。另一种方法是在防火墙中为此类协议使用代理。但这样就会丧失端到端的安全性。这些发展导致有人宣布防火墙将不能作为安全结构的部件，安全服务将从网络转回到终端主机。个人防火墙正在使网络通信上的访问控制转移到终端系统。

## 3.5.4　入侵检测

密码学机制和协议是预防攻击的领域。虽然它们可很好地防御攻击，但在现实中，这些攻击也不易实现。新型的攻击方式已经出现，如拒绝服务攻击。防火墙这样的边界网络安全设备主要用来阻止来自外部网络的攻击，但它们有时做不到。防火墙可能会出现配置错误，密码可能被窃取，新的攻击类型也可能出现。而且，这些设备不能检测攻击发生的时间。因此，为了检测网络攻击，可以配置入侵检测系统(IDS，Intrusion Detection System)。

入侵检测的作用是对网络进行监测，从而提供对各种威胁的实时保护。入侵检测系统在发现入侵后，将采取切断网络连接、记录事件和报警等措施。入侵检测的主要功能有：

- 监测和分析用户和系统的活动；
- 审计系统配置和漏洞；
- 评估系统资源和数据的完整性；
- 识别攻击行为并报警；
- 统计分析异常行为；
- 对系统安全的审计管理，识别违反安全策略的用户活动。

一个 IDS 由一系列收集信息的传感器组成，这些传感器可存在于主机也可存在于网络。这种传感器网络由一个中央控制台管理。分析数据、入侵报告、触发反应都由中央控制台管理。入侵检测有两种途径：误用检测(misuse detection)和异常检测(anomaly detection)。传感器和控制台的通信，特征数据库和产生的日志都应该被保护。而且 IDS 厂家应该提供更新特征库的安全方案。否则，IDS 自身可能受到攻击和操纵。已经有很多利用 IDS 本身的漏洞进行攻击的案例。

### 1. 漏洞评估

漏洞评估用于评价一个网络的安全状态。收集开放的端口、运行的软件包、网络拓扑结构等信息，就可以得到一个不同安全级别的漏洞列表。漏洞评估正如它所基于的技术一样，必须时常更新才能处理新威胁。有关安全组织跟踪安全漏洞，并列出可用补丁。计算机紧急

事故响应小组(CERT，Computer Emergency Response Team)就提供了很好的相关资源，例如CMU(http://www.cert.org)，SANS(http://www.sans.org)和保存了漏洞档案的“安全焦点”(http://www.securityfocus.com)，还有主要的软件和硬件制造厂家的网络站点。

## 2. 误用检测

误用检测(misuse detection)用于寻找攻击特征。攻击特征可以是网络通信中的式样，也可以是日志文件中用于表明可疑行为的活动。这些特征包括在一台敏感主机上登录失败的次数，IP 数据包可以表明一次缓冲溢出攻击的一些位，或者表明一次 SYN 洪泛攻击的某种类型的 TCP SYN 包。入侵检测系统需要考虑安全策略，以及已知的漏洞和记录攻击的数据库。

再者，这些系统的安全性与数据库中的攻击特征库联系在一起。新的漏洞经常被发现和利用，IDS 厂家必须跟上最新的攻击和数据库的更新，用户需要安装更新。数据库已知的漏洞和发掘方法变得大而无用，而且减缓了入侵检测系统的运行。

### 1) 异常检测

统计的异常检测(或者称基于行为的检测)利用统计学技术检测可能存在的入侵。首先，把正常的行为作为基线。在整个操作过程中，将统计分析被监视的数据和估量背离基线的行为。如果超出阈值，就发出警报。这种入侵检测系统不需要知道所监视特定系统的漏洞。基线定义常态。因此，即使没有更新依赖的技术，也可以监测到新的攻击方式。

另一方面，异常检测只检测异常行为。可疑行为不一定会形成一次入侵。在一台敏感主机上的多次试图登录失败可能是由于一次入侵，也可能是管理员忘记密码。攻击也不一定会产生异常。有些攻击者可避开 IDS 监视，不被检测到。特别是当基线被设置为动态和自动调整时，这种情况就会发生。攻击者可逐步躲过那个时间，直到他计划的攻击不再产生任何警报。因此，我们必须关心假阳性(假警报)和假阴性(由于攻击行为在正常行为定义的界限之内，导致攻击无法检测到)。

### 2) 基于网络的入侵检测系统

基于网络的 IDS(NIDS)会在网络通信中查找攻击特征。一个网络适配器以混杂模式运行，实时分析所有经过网络的通信。攻击识别模块把网络数据包作为数据源。有三种常用的技术识别网络攻击特征：式样、表达式或字节码匹配，通过频率或阈值(例如，检测端口扫描活动)，以及与次要事件的相互联系(在商业产品中不多)。Snort 是一种在开源社区中研发的一种流行 NIDS。

## 3. 基于主机的入侵检测系统

基于主机的 IDS(HIDS)从主机的日志文件中查找攻击特征。它也可以验证关键系统文件的校验和以及在规则间隔时间的可执行性。一些产品可用正则表达式推敲攻击特征(例如密码程序的执行和主机文件的改变)。一些 IDS 监听端口的活动，当特殊的端口被访问时产生警报，提供有限的基于网络的 IDS 能力。现在有一种转向基于主机的入侵检测系统趋势。而最有效的入侵检测方法是结合 NIDS 和 HIDS。

由于 IDS 警报已接近实时性，所以 IDS 可用作一个实时响应工具，但自动响应也不可避

免危险。一个攻击者可能欺骗 IDS 响应，但该响应却指向无辜的目标(使用假的源 IP 地址)。由于 IDS 主动的误报，用户会被强迫退出账户。对管理员的 Email 账户重复发送邮件通知会变成拒绝服务攻击。

#### 4. 蜜罐

蜜罐(honeypot)是用于跟踪攻击者和研究、收集黑客活动证据的一种资源。它们模仿真实的系统，其实不包含真正的产品信息。每个被监视到的在蜜罐上的活动都是一次攻击。

蜜罐可在不同层次上用于与攻击者的交互中。低层次交互的蜜罐模仿操作系统基本的服务。在这种蜜罐之上攻击者能做的事很少，蜜罐所记录的攻击性行为就很有限。而且，攻击者可能会很快地识别蜜罐的真相，然后逃走。现在也有很多检测蜜罐的工具。蜜罐模仿得越逼真，更多的行为就会被监视到。高层次交互的蜜罐提供真实的服务，但提供假的数据。攻击者与蜜罐交互得越多，危险越大，因为攻击者会错误地把蜜罐当作台阶去攻击其他机器。

### 3.5.5　安全扫描

安全扫描技术是一类重要的网络安全技术，安全扫描技术与防火墙、入侵检测系统互相配合，能有效提高网络的安全性。安全扫描技术的基本原理是采用模拟黑客攻击的方式对目标可能存在的已知安全漏洞进行逐项检测，以便对工作站、服务器、交换机、数据库等各种对象进行安全漏洞检测。安全扫描技术主要分为两类：主机安全扫描技术和网络安全扫描技术。主机安全扫描技术主要针对系统中不合适的设置和脆弱的口令，以及针对其他与安全规则抵触的对象进行检查等，主机扫描的目的是确定在目标网络上的主机是否可达；网络安全扫描技术则是通过执行一些脚本文件模拟对系统进行攻击的行为并记录系统的反应，从而发现其中的漏洞。端口扫描技术和漏洞扫描技术是网络安全扫描技术中的两种核心技术，并且广泛运用于当前较成熟的网络扫描器中。

通过安全扫描，系统管理员能够发现所维护的 Web 服务器的各种 TCP/IP 端口的分配、开放的服务、Web 服务软件版本和这些服务及软件呈现在 Internet 上的安全漏洞。安全扫描技术也是采用积极的、非破坏性的办法来检验系统是否有可能被攻击崩溃。它利用了一系列的脚本模拟对系统进行攻击的行为，并对结果进行分析。这种技术通常用来进行模拟攻击实验和安全审计。安全扫描系统应具有以下功能：协调其他安全设备；跟踪用户在系统中的行为；报告和识别文件改动；纠正系统的错误设置；识别正在受到的攻击；减轻系统管理员搜索黑客行为的负担；为制定安全规则提供依据。

#### 1. 安全扫描过程

一次完整的网络安全扫描可分为 3 个阶段。第 1 阶段，发现目标主机或网络。第 2 阶段，发现目标后进一步搜集目标信息，包括操作系统类型、运行的服务及服务软件的版本等。如果目标是一个网络，还可以进一步发现该网络的拓扑结构、路由设备及各主机的信息。第 3 阶段，根据搜集到的信息判断或者进一步测试系统是否存在安全漏洞。按照扫描过程来分，扫描技术又可以分为 4 大类：ping 扫描技术、端口扫描技术、操作系统探测技术、如何探测访问控制规则以及已知漏洞的扫描技术。

ping 扫射用于网络安全扫描的第 1 阶段，可以帮助识别系统是否处于活动状态；操作系统探测、如何探测访问控制规则和端口扫描用于网络安全扫描的第 2 阶段，其中操作系统探

测是对目标主机运行的操作系统进行识别，如何探测访问控制规则用于获取被防火墙保护的远端网络的资料，而端口扫描是通过与目标系统的 TCP/IP 端口连接，查看该系统处于监听或运行状态的服务；漏洞扫描通常是在网络安全扫描第 3 阶段，是在端口扫描的基础上，对得到的信息进行相关处理，进而检测出目标系统存在的安全漏洞。

ping 扫描的目的就是确认目标主机的 IP 地址，即扫描的 IP 地址是否分配了主机。对没有任何预知信息的黑客而言，ping 扫描是进行网络扫描及入侵的第一步，也是必不可少的一步；对已经了解网络整体 IP 划分的网络安全人员来讲，也可以借助 ping 扫描，对主机的 IP 分配有一个精确的定位。大体上，ping 扫描是基于 ICMP(Internet Control Message Protocol)协议的，因此把发现目的网络或主机的一类基于 ICMP 协议的扫描称为 ping 扫描。ping 扫描的主要思想就是构造一个 ICMP 包，发送给目标主机，从目标主机生成的响应来进行判断。

ping 扫描能确定目标主机的 IP 地址，随后可以通过端口扫描，探测主机所开放的端口。因为端口扫描通常只做最简单的端口联通性测试，不执行进一步的数据分析，因此比较适合进行大范围的扫描：对指定 IP 地址进行某个端口值范围的扫描，或者指定端口值对某个 IP 地址段进行扫描。然后基于端口扫描的结果，进行操作系统探测和弱点扫描。端口扫描技术发展到现在可细分为许多类型，按照端口连接的情况，端口扫描可分为 TCP 连接扫描、半边接扫描和秘密扫描。其中TCP 连接扫描是端口扫描最基础的一种扫描方式。TCP SYN 扫描在扫描过程中没有建立完整的 TCP 连接，这与 TCP 连接扫描不同，故又称为半连接扫描。秘密扫描包含有 TCP FIN 扫描、TCP ACK 扫描等多种扫描方式。其他端口扫描技术包含 UDP 扫描和 IP 分段扫描等。

### 2. 端口扫描技术

#### 1) TCP 全连接扫描

TCP 全连接扫描就是和目的主机建立一个 TCP 连接，而目的主机的 log 文件中会生成记录。扫描主机通过 TCP/IP 协议的三次握手与目标主机的指定端口建立一次完整的连接。连接由系统调用连接指令开始。如果端口开放，则连接将建立成功；否则，若返回-1，则表示端口关闭。目标主机的指定端口以 ACK 响应扫描主机的 SYN/ACK 连接请求，这一响应表明目标端口处于监听(打开)的状态。如果目标端口处于关闭状态，则目标主机会向扫描主机发送 RST 的响应。

#### 2) TCP SYN 扫描

全连接扫描中扫描主机和目的主机建立一个 TCP 连接，目的主机的 log 文件中会生成记录。与全连接扫描相对应，半连接扫描也称为 TCP SYN 扫描。TCP SYN 扫描(半开放扫描)是利用 3 次握手的弱点来实现的。通过向远程主机的端口发送一个请求连接的 SYN 数据报文，如果没有收到目标主机的 SYN/ACK 确认报文，而是 RST 数据报文，就说明远程主机的这个端口没有打开。而如果收到远程主机的 SYN/ACK 应答，则说明远程主机端口开放。在收到远程主机的 SYN/ACK 后，不再做 ACK 应答，这样 3 次握手并没有完成，正常的 TCP 连接无法建立，因此这个扫描信息不会被记入系统日志，不会在目标主机上留下记录。

#### 3) 秘密扫描技术

秘密扫描是一种不被审计工具所检测的扫描技术。很多防火墙和路由器对半连接扫描都有了相应措施，这些防火墙和路由器会对一些指定的端口进行监视，将对这些端口的连接请求全部进行记录。秘密扫描能躲避 IDS、防火墙、包过滤器和日志的审计，从而获取目标端口的开放或关闭的信息。这种技术不包含标准的 TCP 三次握手协议的任何部分，所以无法被记录下来，比 SYN 扫描隐蔽得多。秘密扫描有 TCP FIN 扫描、TCP ACK 扫描、NULL 扫描、XMAS 扫描和 SYN/ACK 扫描等。

TCP FIN 扫描的原理是扫描主机向目标主机发送 FIN 数据包来探听端口，若 FIN 数据包到达的是一个打开的端口，数据包则被简单地丢掉，并不返回任何信息，当 FIN 数据包到达一个关闭的端口，TCP 会把它判断成是错误，数据包会被丢掉，并返回一个 RST 数据包。这种方法与系统的 TCP/IP 实现有一定关系，并不是可以应用所有系统上，有的系统不管是否打开，都回复 RST，这种扫描方法不适用。

TCP ACK 扫描主要用来探测过滤性防火墙的过滤规则，无论目标端口的状态如何，如果发送 ACK 报文，就只能收到 RST 响应报文。但是对于防火墙，如果端口被过滤，要么收不到报文，要么收到 ICMP(目标不可达)，相反，如果没有被过滤，则收到有关 RST 报文。

XMAS 和 NULL 扫描是秘密扫描的两个变种。XMAS 扫描将 TCP 包中 ACK FIN、RST、SYN、URG、PSH 标志位置 1，而 NULL 扫描关闭所有标记。这些组合的目的是为了通过所谓的 FIN 标记监测器的过滤。目标端口开放的情况下，目标主机将不返回任何信息，若目标端口关闭，则目标主机将返回 RST 信息。

SYN/ACK 扫描中，扫描主机不向目标主机发送 SYN 数据包，而先发送 SYN/ACK 数据包。目标主机将报错，并判断为一次错误的连接。若目标端口开放，目标主机将返回 RST 信息，若目标端口关闭，目标主机将不返回任何信息，数据包会被丢掉。

秘密扫描通常适用于 UNIX 目标主机，除了少量的应当丢弃数据包却发送 reset 信号的操作系统(包括 CISCO、BSDI、HP/UX、MVS 和 IRIX)外。在 Windows95/NT 环境下，该方法无效，因为不论目标端口是否打开，操作系统都发送 RST，与 SYN 扫描类似，秘密扫描也需要自己构造 IP 包。

#### 4) 其他扫描技术

其他扫描技术主要有 UDP 扫描、IP 分段扫描和代理扫描。

UDP 是面向无连接不可靠的数据包协议，建立在 UDP 协议基础上的 UDP 端口扫描是不可靠的。UDP 扫描主要是 UDP ICMP 端口不可到达扫描。从 UDP 协议可知，如果 UDP 端口打开，则没有应答报文；如果端口关闭会有 ICMP 报文(端口不可达)。这样，只须构造一个 UDP 报文，观察响应报文就可知道目标端口的状态。虽然用 UDP 提供的服务不多，但是有些没有公开的服务很可能是利用的高端口服务。如果需要证实有这样的服务，就可以利用这种扫描技术。

在 IP 分段扫描中，主机并不直接发送 TCP 探测数据包，而将数据包分成两个小的 IP 段，这样就将一个 TCP 头分成好几个数据包，从而过滤器就很难检测到，扫描就可以在不被发现的情况下进行。但是对 IP 分段扫描，一些程序在处理小数据包时会有些麻烦，并且不同的操

作系统在处理这个数据包时，也通常会出现问题。

代理扫描，它基于文件传输 FTP 协议。FTP 服务器可发送文件到互联网的任何地方，FTP 代理扫描难以跟踪，许多扫描器利用这些特点实现 FTP 代理扫描。FTP 端口扫描中，扫描主机和目标主机之间有一个能支持代理选项的 FTP 服务器、通过代理 FTP 服务器与扫描主机及目标主机的连接是否能实现来判断目标主机的某些端口的状态，由于有代理服务器的参与，FTP 代理扫描难以跟踪，可以实现基于 FTP 协议的代理端口扫描。

### 3. 漏洞扫描技术

系统安全漏洞也称为系统脆弱性(vulnerability)，一般简称漏洞。网络入侵的过程一般是利用扫描工具对要入侵的目标进行扫描，找到目标系统的漏洞或弱点，然后进行攻击。对于系统管理员来说，网络安全的第一步工作也应该是利用扫描工具扫描系统，发现系统的漏洞和弱点后采取相应的补救措施。漏洞是计算机系统在硬件、软件、协议的设计与实现过程或在系统安全策略上存在的缺陷和不足。漏洞的产生主要是由于程序员不正确和不安全的编程引起的。漏洞扫描实际上是一个逐步探测漏洞的过程，也是一个漏洞扫描器最核心的部分。漏洞扫描主要通过以下方法来检查目标主机是否存在漏洞：

1) 在端口扫描后得知目标主机开启的端口及端口上的网络服务，将这些相关信息与网络漏洞扫描系统提供的漏洞库进行匹配，查看是否存在满足匹配条件的漏洞。

2) 通过模拟黑客的攻击手法，对目标主机系统进行攻击性的安全漏洞扫描，如测试弱势口令等。若模拟攻击成功，则表明目标主机系统存在安全漏洞。基于网络系统漏洞库，漏洞扫描大体包括 CGI 漏洞扫描、POP3 漏洞扫描、FTP 漏洞扫描、SSH 漏洞扫描、HTTP 漏洞扫描等。这些漏洞扫描将扫描结果与漏洞库相关数据比较得到漏洞信息。漏洞扫描还包括没有相应漏洞库的各种扫描，比如 Unicode 遍历目录漏洞探测、FTP 弱势密码探测、OPEN Relay 邮件转发漏洞探测等，这些扫描通过使用插件(功能模块技术)进行模拟攻击，测试出目标主机的漏洞信息。

3) 基于网络系统漏洞库的漏洞扫描工作时，首先探测目标系统的存活主机，对存活主机进行端口扫描，确定系统开放的端口，同时根据协议指纹技术识别出主机的操作系统类型。然后根据目标系统的操作系统平台和提供的网络服务调用漏洞资料库中已知的各种漏洞进行逐一检测，通过对探测响应数据包的分析判断是否存在漏洞。基于网络系统漏洞库的漏洞扫描的关键部分就是它所使用的漏洞库通过采用基于规则的匹配技术，即根据安全专家对网络系统安全漏洞、黑客攻击案例的分析和系统管理员对网络系统安全配置的实际经验，可以形成一套标准的网络系统漏洞库，然后在此基础之上构成相应的匹配规则，由扫描程序自动进行漏洞扫描工作。漏洞库信息的完整性和有效性决定了漏洞扫描系统的性能，漏洞库的修订和更新的性能也会影响漏洞扫描系统运行的时间。因此，漏洞库的编制不仅要对每个存在安全隐患的网络服务建立对应的漏洞库文件，而且应能满足前面所提出的性能要求。

4) 在基于功能模块技术的漏洞扫描中，插件是由脚本语言编写的子程序，扫描程序可以通过调用它来执行漏洞扫描，检测出系统中存在的一个或多个漏洞。添加新插件就可以使漏洞扫描软件增加新功能，扫描出更多漏洞。插件编写规范化后，甚至用户自己都可以用 perl、C 或自行设计的脚本语言编写的插件来扩充漏洞扫描软件的功能。这种技术使漏洞扫描软件

的升级维护变得相对简单，而专用脚本语言的使用也简化了编写新插件的编程工作，使漏洞扫描软件具有强的扩展性。

除了基于网络的漏洞扫描技术外，还有基于主机的漏洞扫描技术。基于主机的漏洞扫描，就是通过以 root 身份登录目标网络上的主机，记录系统配置的各项主要参数，分析配置的漏洞。通过这种方法，可搜集到很多目标主机的配置信息。在获得目标主机配置信息的情况下，将其与安全配置标准库进行比较和匹配，凡不满足者即视为漏洞。通常在目标系统上安装了一个代理(agent)或服务(services)，以便能够访问所有的文件与进程，这也使得基于主机的漏洞扫描器能够扫描更多漏洞。

#### 4. 操作系统探测技术

操作系统探测技术分为主动探测技术和被动探测技术。主动探测技术通过向目标系统发送数据，促使其做出响应，然后提出和分析响应数据的特征信息，以推测目标系统的操作系统类型；被动探测技术不主动激发目标系统的响应，而通过网络嗅探来截获目标系统发出的数据包，从中提取和分析特征信息来获得操作系统的类型。被动探测方式的隐蔽性较强，而主动探测方式的手段更显丰富，更具有技巧性。常用的主动探测手段有：

- 查询标识信息，最简单的情况是目标系统将自身操作系统的有关信息作为服务的内容提供给外界。例如，在系统提供的服务中，给用户返回的信息中顺便告知操作系统的类型。
- 二进制文件分析，通过收集、分析目标系统上的二进制文件，同样可以获得其操作系统信息。
- ICMP 信息探测，Internet 控制消息协议(ICMP)是 TCP/IP 协议栈中的一个组成部分，主要用于在 IP 主机、路由器之间传递控制消息。目标系统响应 ICMP 命令时返回的数据，常常反映出其操作系统的特征。例如，微软 ICMP 请求报文的有效载荷中包含字母，而 Solaris 和 Linux 的 ICMP 请求报文的有效载荷中包含了数字和符号。

被动探测抓取目标主机发送出来的 TCP 报文，通过分析报文中的有关字段，获取操作系统的特征信息。常用于操作系统类型分析的字段有：

- 生存期(TTL)，TTL 表明该包在网络中的生存时间。不同操作系统的默认 TTL 不同，从而由 TCP 报文中的 TTL 推测远程操作系统类型。例如，许多操作系统的默认 TTL 小于 30，Linux 和 FreeBSD 的默认 TTL 值被设置为 64。当被抓取包中的 TTL 大于 30，有理由推测其发送者可能就是 Linux 或 FreeBSD。
- 滑动窗口大小，在 TCP 协议中不同操作系统设置的滑动窗口大小是不同的，因此也可以作为探测依据。例如，Cisco 路由器和微软的操作系统发送的 TCP 报文中的滑动窗口在一次会话过程中是经常改变的。Linux、FreeBSD 和 Solaris 系统并没有实现滑动窗口算法，它们发送的 TCP 报文在一个会话过程中是维持不变的，而且就是某个默认常数。
- 分片允许位(DF)，许多操作系统会使用 DF 位置，但是 SCO 和 ORENBSD 不会使用 DF 标志。
- 服务类型(TOS)，当 ICMP 端不可到达时，目的主机回送的包中经常使用 0，但是 Linux 用的是 0xC0，这不是标准的 TOS，而是一个未使用优先域(AFAIK)的一部分。

- 初始化序列号(ISN)，不同的操作系统会为 TCP 连接选择不同的初始化序列值。例如，新版本的 Solaris、IRIX、FreeBSD、Digital、Unix 使用的是一个随机增量，Windows 采用一个“时间相关”模型，每隔一段时间在 ISN 会被加上一个小常数。

随着人们信息安全意识的增强，信息安全技术的发展以及网络复杂环境的增加，对操作系统的准确探测也会变得更困难。要提高远程操作系统探测的准确度和效率，不仅需要对操作系统远程探测技术进行更深入的研究，还需要结合其他方法进行综合探测。

## 3.6　本章小结

本章介绍了数据的加密方法和加密原理，以及密钥的安全和管理方法。在使用一个安全系统时，需要检查和验证请求服务的用户身份，这时用到了 3.2 一节中的身份验证技术。当用户成功登录后，需要通过访问控制技术对文件资源进行保护和管理。3.4 一节的安全保障技术是判断信息系统可信度的基础，它能够检验并且提高系统可信度。网络安全部分主要通过网络威胁模型引出网络协议安全、防火墙技术、入侵检测和安全扫描方面的知识。

# 第4章　代码安全静态分析

## 本章导读

程序静态分析(Program Static Analysis)是指在不运行代码的情况下对代码进行评估的过程。静态分析非常强大，因为它允许对多种可能性进行快速考量。静态分析尤其适合用于安全方面的检查，许多隐蔽的安全问题可以通过对大量代码进行可靠而详尽的评估而被发现。本章将详细介绍程序静态分析的基础知识。

应掌握的知识要点：

- 静态分析的概念；
- 静态分析的局限性；
- 通过静态分析能够解决的问题；
- 执行代码审查；
- 安全审查；
- 静态分析的过程；
- 静态分析中常见的问题；
- 缓冲区溢出。

## 4.1　静态分析

### 4.1.1　静态分析的概念

程序最重要的属性之一是正确性。长期以来，专家学者们针对如何保证程序的正确性、尽可能发现程序中各种潜在的错误这个问题提出了各种各样的代码分析方法或技术，其中程序静态分析是最普遍也是最常用的一种。

程序静态分析(Program Static Analysis)是指在不运行代码的方式下，通过各种分析工具对程序代码进行扫描并做出评估的过程。

静态分析主要具有以下特点：

1) 不实际执行程序，只是通过对代码的静态扫描对程序进行分析；
2) 执行速度快、效率高。

### 4.1.2　静态分析的局限性

对入行不久的程序员来说，对安全编程所需的知识缺乏理解会导致其所编写的代码不符合安全规范而引发安全问题。另外，即使是经验丰富的程序员也无法完全避免在代码编写的

过程中出现或多或少的拼写错误，这类简单错误有时也可能引发安全问题。程序员完全不知道攻击者会以怎样的方式来尝试利用某段代码，因此即使是最细微的错误也可能是相当重大的安全隐患。树立了这种认识，我们就意识到程序静态分析的重要性，从而通过进一步学习感受其相当强大的安全问题识别能力。

静态分析工具能够完整而客观地进行程序检测，而不管编码人员是否了解哪些代码是与安全“相关的”。与采用人工校对的方式相比，这种静态分析方法显得更加可靠和中立。

静态分析可在开发早期发现错误，甚至是程序第一次运行之前就可以发现，这不但降低了修补漏洞的花销，而且这种迅速反馈机制也有助于指导程序员的工作。程序员也有机会修改他们之前没有注意到但却很可能发生的错误。静态分析工具所使用的攻击方法与代码构建相关的信息充当了知识转移的手段。

当一种新的攻击出现时，静态分析工具可以迅速地对相关代码进行复查，同时分析该新型攻击能否对代码构成威胁。在被发现之前，某些安全缺陷已在程序中存在很久了，静态分析工具有能力针对新发现的缺陷类型对遗留代码进行检查。

从上面的叙述中我们可以更清楚地看到程序静态分析的安全问题识别能力，但事实证明，程序静态分析方法并非十全十美，它在代码安全问题检测的大框架下仍然有自己的局限性。其中最普遍的局限性就是：由于代码静态分析是通过对程序扫描找到匹配某种规则模式的程序代码以发现其中存在的问题，这样有时会出现将某些正确代码错判为缺陷的问题，因此在对代码检查的过程中会产生太多无用信息，尤其是产生太多误报。过多的误报会带来很多困难。过多的误报信息不仅会增加测试人员和程序员的劳动力，而且，查看冗长的误报列表还可能遗漏掉隐藏在列表中的真实漏洞信息。

尽管误报难以避免，但从安全的角度看，漏报才是最严重的局限性。漏报是指程序中存在缺陷，但安全分析工具却没有检测出该缺陷。误报的代价是对分析报告的审查会花费更多时间，但漏报的代价却更为严重。不仅要为与代码中存在漏洞相关的后果负责，而且程序会一直存在于安全的假象中，这种假象源于分析工具表象的检查结果。

大多数静态分析工具都会产生一些误报或者漏报的情况，而一个静态分析工具在误报和漏报之间进行平衡的侧重不同，往往预示着其用途的不同。用于检测程序中普通 Bug 的静态分析工具和专用于检测安全缺陷的静态分析工具相比，前者通常会力求产生少量的误报而更愿意接收漏报的情况，而后者在漏报时付出的代价可能会很高，所以它通常会产生更多的误报而尽量减少漏报的情况。

### 4.1.3　静态分析方法

静态分析的使用比大多数人所了解的要更广泛，在这一节，我们来看看静态分析的方法，静态分析所拥有的方法通常包括：类型检查、风格检查、程序理解、程序验证、属性检查、Bug 查找和安全审查，本节将重点介绍它们。

#### 1. 类型检查

类型检查是最普遍的静态分析形式，也是大多数程序员熟悉的形式，但很多程序员并未给予足够关注，主要原因是类型的规则通常都是由编程语言定义，并由编译器强制执行，程序员无权决定什么时候执行分析，或者分析如何运作。即使这样，类型检查依旧是一种基本

的静态分析，它将消除代码中存在的类型错误。

类型检查对代码中错误的检测能力是有局限的，另外，与其他所有静态分析技术一样，类型检查也存在误报和漏报的问题。

例如：

```
Object[] obj = new String[1];
obj[0] = new Object();
```

Java 语句能通过类型检查和编译，但在运行时将会失败，Java 中的数组是可变的，类型检查工具允许一个 Object 数组变量保存一个指向 String 数组的引用，但在运行时，Java 不允许这个 String 数组保存对 Object 类型对象的引用。类型检查工具不会对代码有任何疑问，但代码运行时将抛出 ArrayStoreException 异常，这表示类型检查的一次漏报。

### 2. 风格检查

风格检查也是静态分析工具的一种。由于许多程序员认为的好风格都具有强烈的个人色彩，因此大多数风格检查程序在其所执行的规则集中都十分灵活。由风格检查程序所显示的错误常只是影响代码的可读性和可维护性，而不是程序运行过程中会发生的某种错误。随着时间的推进，很多编译器也已经实现了可选的风格检查功能。

在一个大型编程项目的中途很难采用一个风格的检查工具，因为对于“正确”的风格，不同程序员很可能已经形成了自己不同的看法。在项目启动后，仅为减少风格检查时的输出而重新检查一遍程序以调整代码风格，这不会实现多少实际利益。因此，在项目的开始采用风格检查是最容易的。

目前，有许多开源的和商业的风格检查程序可供使用。其中，迄今为止最著名也是最古老的工具是 Lint。原来许多由 Lint 所执行的检查，现在已经并入到当前流行的编译器所提供的各种警告级别中，但是“类 Lint”这个词则成了用于描述风格检查程序的歧视性术语。Java 风格检查通常使用开源 PMD，因为该程序可以很容易地选择你想要遵循的风格规则，并且添加自定义规则也同样简单。

### 3. 程序理解

程序理解工具能够帮助程序员理解代码库中的海量代码。集成开发环境通常都包含程序理解模块。例如查找本方法的所有应用或找出全局变量的声明等。更高级一点的分析可支持自动进行程序重新分解组合的功能，例如对变量重命名或将一个函数分割成多个函数。

高水平的程序理解工具将努力帮助程序员理解程序运作的原理。某些工具试图根据对现实的分析，反向生成该程序的设计，从而为程序员提供关于此程序的一个大概视图。尽管这只是原始设计本身的一个拙劣的替代品，但它可以有效地帮助程序员搞懂由他人所写的大量程序代码。例如 Fujaba 工具包允许在 UML 图表和 Java 源代码之间来回转换，某些情况下，Fujaba 还可以从其读取的源代码推断出设计模式。

### 4. 程序验证和属性检查

程序验证工具接受一份规格说明及其对应的代码，而后试图证明该代码完美实现了此规格说明。如果这份规格说明完整描述了程序所要做的所有事情，那么该程序的验证工具就可执行等价检查以确认该程序的代码与这份规格说明完全匹配。程序员通常很少能有一份用来

进行等价检查的规格说明，而且创建一份这样的规格说明所需要做的工作最终将比编写代码的工作量还要大，因此，通常人们不会进行此类正式验证。另外，从历史上看，等价检查工具还不能解决任何大型程序遇到的问题。

验证工具将根据只描述部分程序行为的部分规格说明对软件进行检查，这种检查方式有时称为属性检查，多数属性检查工具或者通过应用逻辑推论，或者通过执行模型检查来实现其功能。

很多属性检查工具将其重点放在临时安全属性上。临时安全属性规定了一系列有序的事件，而在程序中绝对不能发生这些事件，例如“一个内存位置，当其释放之后，就不应该再被读取”。大多数属性检查工具都允许程序员编写自己的规格说明来检查程序特定的属性。

当属性检查工具发现代码有可能与规格说明不匹配时，通常会通过报告一个反例来向用户说明它的发现。只要有问题存在，属性检查工具就会报告这个问题，那么我们就称这个属性检查工具是健全的，遵守了相应的规格说明。即这个工具永远不会出现漏报的情况。大多数声称为健全的工具都会要求被评估的程序符合一些特定条件。某些工具不允许使用函数指针，还有一些不允许使用递归，或者假定两个指针永远不会互为假名(即不会指向同一个内存位置)。在学术环境中，健全性是一个十分重要的特性，因为这方面任何程度的不足都可能被贴上“没有原则”的标签。但对于现实中的大量代码来说，几乎不可能满足属性检查工具所规定的条件，因此其健全性保证就没有什么意义。由于这种原因，从业界人士的角度看，健全性很少成为一种对此类工具的需求。

由于追求健全性或者其他复杂的原因，属性检查工具也可能产生误报。在误报的情况下，反例中将包含一个或多个实际上并不会发生的事件。

### 5. Bug 查找

Bug 查找工具的目的，既不是像风格检查程序那样抱怨格式化问题，也不是像程序验证工具那样执行完全而彻底的程序和规格说明之间的比较。它是用来指出程序中存在的以超出程序员设想的方式运行的那些代码。大多数 bug 查找工具的使用都十分简单，因为它们都遵守一套默认规则，用来描述程序中常常预示含有 bug 的一些模式。

比较高级的 bug 查找工具能够通过代码本身推测出需求从而扩增其内建模式。有些 bug 查找工具使用了与属性检查工具一样的算法，但是 bug 查找工具追求的是产生较少的误报，哪怕会产生较高的漏报率。理想的 bug 查找工具就反例而论是健全的，也就是说，当该工具生成一份 bug 报告时，其随附的反例总是表现为一个程序中切实可行的时间序列。

### 6. 安全审查

以安全为中心的静态分析工具使用了许多在其他工具中使用的技术，但对于其目标的关注意味着会以不同的方式来应用这些技术。

最早的安全工具非常贴近于风格检查工具，它们所指出的问题不一定能引起安全问题，但是，它们表示了一种日益提高安全关注的思考，这些工具一次次地因为较高的误报率而被指控，这是因为人们试着将这些工具的输出作为一份 bug 列表来看待，而不是将其作为代码审查期间的一种辅助手段来用。

现代安全分析工具往往更像是一种属性检查程序和 bug 查找程序的混合体，许多安全属

性能被简洁的表达为程序属性，对于一个属性检查程序来说，搜索潜在的缓冲区溢出漏洞可以当做是检查这样的程序属性："程序不会访问被分配内存的边界之外的地址"。从 bug 查找领域，安全分析工具采纳了这样一种观念，即开发人员往往会继续再使用相同的不安全的方法来解决问题，这可说成是一种不安全的惯例。

# 4.2　代码审查中的静态分析

关于如何使用静态分析工具，有很多东西需要了解。或许我们只需知道将静态分析工具作为安全开发过程的一部分来使用。就此而言，辅助进行安全审查的工具和大多数其他类别的软件开发工具相比，存在根本的不同。例如，调试器不要求落实任何组织范围的规划。作为个体时，程序员可以在需要时运行这个调试器，获得结果，并继续其他编程任务。仅仅出于软件安全的考虑很少会产生急需程序员运行调试器的情况。出于这种原因，一个组织需要有一个计划，规定谁将指导安全审查，什么时候要进行审查以及如何基于审查结果来采取行动。由于静态分析工具可显著提高安全审查过程的效率，故应将静态分析工具作为这个计划的一部分。

## 4.2.1　执行代码审查

之所以要开展以安全为中心的代码审查，有多种原因：

- 某些审查人员的出发点是需要找出一些可利用的漏洞，用于证明在安全方面额外投资的正确性。
- 对于每个开始的时候没有安全意识的大型项目，开发团队最终不得不回过头来重新检查一遍代码以进行安全方面的改进。
- 每个项目至少在每次新版本发布期间要接受一次安全审查，以充分考虑要新增的功能特性和正在进行的维护工作。

在这 3 条中，迄今为止，第 2 条要求付出的时间和精力是最多的。改进一个编写时未考虑安全的程序将是一项工作量相当大的任务。对同一段代码，后续审查会相对容易一些。最初的审查很可能会找出许多需要解决的问题。后续的审查发现的问题应该会相对少些，这是因为程序员会在一个更健壮的基础上构建程序。

### 1. 代码审查周期

首先，我们来概要地看一下代码审查周期，然后详细对每个阶段进行讨论。在这个周期中有 4 个主要的阶段：

#### 1) 确立目标

一组定义完善的安全目标将有助于对应该被审查的代码以及应该用于审查这些代码的准则进行优先级排序。这些目标应该来自于你所面对的软件风险的评估。我们常常会听到一些笼统的高级目标，通常会以下面这几句的形式阐述：

- "如果能通过 Internet 访问到它，那么必须在其发布之前对其进行审查。"
- "如果它将处理钱的问题，那么必须至少每年对其审查一次。"

我们在和人聊天的时候也会了解到一些人有更具体的战斗性目标。一个短期的关注点可能来自于以下某个声明：

- “我们不能在下次的一致性审计中出问题。要确保审计员给我们提供的是一张干净健康的清单。”
- “我们要结束各种跨站点执行脚本漏洞的困扰。”

你需要有高级指导来对代码审查目标进行优先级排序。以单个程序为单位由高向低地逐级设置审查的优先级。当你已经按照这种粒度进行划分时，就不要再进一步细分了；至少每次都针对一个完整的程序运行静态分析。如果你认为有部分程序会带来更高风险，那么你可能选择以更详细的方式或更频繁地审查分析结果，但是，尽量要让静态分析工具的结果来引导你的关注点。

当我们问人们他们进行代码审查时在寻找什么的时候，我们听到的最常见的回答：“哦，排在靠前的问题吧？”很糟糕的答案！最大的问题就是最后的这个“？”，如果你不能确定自己找什么，那么你就没有什么机会找到它。此外，“排在靠前”的部分也不是焦点问题。针对排在靠前漏洞而进行的检查也只是部分遵循了支付卡行业(PCI，Payment Card Industry)数据安全标准，但对于你应该查找的那些问题来说，这既不能作为开始，也不能以此为最终目标。如果你需要灵感，那就去检查一下之前对你计划审查的程序或者类似程序所进行的代码审查结果。之前发现的问题会以某种不可思议的方式再次潜回到程序中。对过去的结果进行审查还能为你提供机会来了解自之前的审查之后都有些什么东西发生了改变。

确保审查人员了解所审查代码的功能和用途。关于此代码的设计，如果有一份高级的描述的话，将会有一些帮助。这时，也应该去审查一下与此代码相关的风险分析结果。如果在开始工作之前，审查人员对风险不够了解的话，那么，随着审查的推进，关于什么是可以接受的，什么是不可以接受的，集体的意见会不断演变，所以，这种结果将不是那么理想。那种“我看到它的时候，就会了解这个安全问题”的“即兴”方式不会产生最理想的结果。

#### 2) 运行静态分析工具

头脑中有了审查目标，我们就可以开始运行静态分析工具了。要开始这个过程，需要收集目标代码，对工具进行配置以报告那些带来最大风险的问题，而后禁用无关的检查。本阶段的输出将作为在代码审查期间使用的一组原始结果。

为得到有效结果，你应该能够编译所分析的代码。对于在自己的构建环境中运行的开发团体来说，这不是什么问题，不过，对于间接获得代码的安全团队来说，这确实会是个大问题。所有的头文件在什么地方？你用的是哪个版本的库文件？各种明里暗里的障碍能列出一张很长的清单。这时，你可能面临着采取某种捷径的诱惑。即使是所得到的代码不能编译，静态分析工具往往也至少能产生一些结果。在执行分析之前，使代码进入到可编译状态。如果你习惯于忽略来自静态分析工具的语法解析错误和分析警告，你往往会错过重要的结果。

这时，你还应该添加自定义的规则，从而检测专门针对所分析程序的错误。如果你的组织有一套安全编码指导原则，仔细检查这些原则，并找出可作为自定义规则进行编码的那些条目。默认状况下，静态分析工具不会知道在你的代码所处的上下文环境中，什么会构成安全违规。通过根据实际环境而对工具进行定制，将获得很好的机会来极大地提高该工具所得

出结果的质量。

在前面进行的手工代码审查过程中发现的错误在这里也特别有用。如果之前识别出的错误能表述为违反了某种程序不变式，那么就可以编写一个规则来检测类似的情况。随着时间的推移，这个规则集将作为一种制度上的备忘形式，从而防止前面出现的安全失误重复出现。

#### 3) 审查代码(使用静态分析工具的输出)

现在是审查代码的时候了。仔细检查静态分析结果，但不要将自己局限于仅仅是分析结果。要让分析工具发现潜在的问题，但不要被它在其他问题上误导，而这些问题原本可以通过自己对代码的检查来发现。我们通常可以找出与分析工具所报告问题相关的其他 bug。这种“邻近效应”源于这样的事实，即静态分析工具经常会在敏感操作附近感到困惑，这时就会报告一个问题。尽管可能原因不尽相同，但能让工具产生困惑的代码往往也会让程序员觉得困惑。仔细检查所有静态分析结果，不要仅停留在高优先级的警告。如果结果清单比较长，将其分割，这样可以由多个审查人员共同来完成这项工作。对单一问题进行审查是对工具报告这个问题时所做的假设进行验证。采取缓解措施能够防止这段代码出现漏洞吗？非受信数据源是否确实不可信？由工具假设的情景实际上是否切实可行？如果你正在审查其他人的代码，你不能够回答所有这些问题，而应该与代码的作者或所有者合作。某些静态分析工具会让结果的共享非常容易，这将简化这个过程。

协力完成的审计是一种对等审查的形式，结构化的对等审查时一种经过证明的技术，可用于识别所有类型的缺陷。为执行关注安全的对等检查，最好在审查团队中有一个安全专家。对等审查和静态分析都是补充技术。当我们执行对等审查的时候，我们通常会设置一位审查人员来负责仔细检查工具的输出。

代码审查结果可采取许多种形式：输入到数据库中的 bug、适合程序员和管理目的的使用的一份形式化报告、软件安全跟踪系统中的条目或是一份送给程序员的非正式任务清单。无论其形式是什么，都要确保能够得到永久保存，以便在下一次代码审查期间可以使用这些结果。对每个问题的反馈应该包含一份对问题的详细解释、一份对该问题所带来的风险的评估以及相关部分的安全策略和风险评估文档参考。这种对审查结果的持久收集还适用于另一种用途，即作为安全培训的信息。你可以使用审查结果将培训聚焦于与代码紧密相关的那些实际问题和话题。

#### 4) 进行修补

有两种因素在左右程序员对来自安全审查反馈进行响应的方式：

- 安全是他们的事情吗？如果做好安全是发布他们代码的先决条件，那么安全就是他们的事情。不这样的话安全就无法贯彻，因为这项工作和添加新功能、打补丁以及按时将代码发布之类的工作是竞争性的，做了这个，其他的就会受到影响。
- 他们理解这份反馈吗？理解安全问题需要进行安全培训。此外，还需要将这种反馈以一种能被理解的方式来编写。源自代码审查的结果不像故障测试用例那样具体，所以此结果需要对所涉及的风险给出更完整的解释。

在开发生命周期中，如果安全审查开展得足够早的话，那么这时就应该对来自安全审查的反馈进行响应。是不是围绕某个具体的模块或者某个具体的功能而出现大堆的问题呢？如果是的话，这时候就应该回溯并查找设计相关的问题，这样可以减轻这些问题的严

重性。如果没有这种情况，你可能会发现，通过额外安全培训的形式，问题得到了最好、最持久的解决。

当程序员解决了审查所识别出的问题时，相应的修复必须进行检验。所采取的验证形式取决于所做更改的性质。如果所涉及的风险较大而所做的更改也不小，那么就应该返回到审查阶段并再次进行代码的检查。

#### 2. 避开可利用性陷阱

安全审查本不应该忙于创建一些华而不实的漏洞利用手段，但是事实上这种现象太常见了，安全审查团队往往限于漏洞利用手段的开发之中。要理解为什么会这样，先来看一下在安全审查期间，一段代码可能得到的 3 种判定：

- 明显是可利用的；
- 不能确定是否安全；
- 明显是安全的。

在这些情况之间不存在清晰的界线，它们形成一种光谱状的连续分布。这种分布的最末端比中间部分的麻烦要少，明显可利用的代码要进行修补，而明显安全的代码可以不用去管它。对于中间的情况，也就是那些不能确定是否安全的代码，是比较棘手部分。代码之所以会不清楚，是因为其逻辑难以跟踪，因为难以判断在何种情况下会调用这段代码，或者是因为难以了解攻击者会怎样来利用这个问题。

而危险就在于审查人员对待这种不确定代码的方式。如果由审查人员来负责在对代码修补之前证明这段代码是可利用的，那么，安全审查人员最终将误入歧途而忽略可利用的 bug。在程序员说“除非你证明它是可利用的，否则我不会解决这个问题”的时候，你正面对着可利用性陷阱。

可利用性陷阱是危险的，这有两种原因。第一，开发漏洞利用手段是一件消耗时间的工作。与其将你的时间花在开发漏洞利用手段上，还不如把它花在查找更多问题上。第二，开发漏洞利用手段是一种与其他工作不大一样的独特技能。如果你不能开发漏洞利用手段的话，会怎么样呢？意味着这个缺陷就不可利用了吗？还是意味着你不知道该用哪些技巧来利用它呢？

不要落入可利用性陷阱，而要解决这些 bug。如果一段代码不是明显安全的，设法让它变得明显安全。有时这种方式会带来冗余的安全检查。有时这会引起一种意见，要求提供可证实的方法来判断这段代码是好的。而有时，这也会在代码中插入一个可利用的漏洞。当没有提出错误时，程序员不会总是狂热地想要修改代码，因为任何更改都会带来引入新 bug 的可能性。但另一个方面——发布带漏洞的代码，程序员更不愿意这样。

除了被忽略的 bug 最终可能导致的风险之外，一种新的漏洞利用方式就是可能性：仅仅证明 bug 的可能性，而不需要证明它是可利用的，这样，就有可能导致某个公司的声誉受损。软件公司有时会发现，即使所有的迹象都表明某个缺陷不是可利用的，他们也需要发布相应的安全补丁。

### 4.2.2　开发过程中的安全审查

将安全集成到软件开发过程中，这说起来容易，但是如果程序员习惯于忽视安全性的

话，这将经历一次艰难的转变。评估和选择静态分析工具可能是软件安全的活动中最容易的部分。工具可使得程序员能更高效地处理软件安全问题，但只有工具不能解决任何问题。换句话说，静态分析应该作为安全的开发生命周期的一部分，而不是用其替代安全的开发生命周期。

任何成功的安全的活动都要求程序员接受一种理念，那就是安全非常重要。在传统的层次化组织中，这通常意味着来自管理层的对安全重要性的宣言，紧跟着一个或多个管理层发出的信号，说明安全确实应该认真对待。

接受静态分析工具已逐渐成为推动安全的一部分。出于这一原因，静态分析和安全改进这两个过程往往交叉进行。这一节主要着力解决与工具采用相关的障碍。

我们以前见过的所有软件的开发组织，至少都有那么一点点混乱，而要改变一个混乱系统的行为并不是什么好办的事。乍一看来，采用一个静态分析工具看上去不会有太大的问题。获得这个工具，运行之，解决问题，而后你就搞定了。真的是这样吗？错。仅仅依靠一个新工具就期望对安全状态进行改变，这是不切合实际的。

根据我们的经验，要成功采用一个静态分析工具，有 3 个大问题必须回答。组织的大小，以及开发过程的类型和成熟度，在这些答案中都有着重要的影响。所以这些问题都没有通用的答案，所以我们只能考虑每个问题可能的答案。这 3 个问题如下。

#### 1. 谁会运行此工具？

实际上，在理想情况下谁会运行这种工具都没有什么关系，不过，从实践角度考虑，这成了一个非常重要的问题，比如，对代码的访问就需要考虑一个因素。对于谁会运行此工具，许多组织有两种明显的选择：安全团队或者程序员。

##### 1) 安全团队

要使其正常运作起来，你必须确保你的安全团队具有适当的技能——简言之，你希望安全人员能有一点软件开发的技能。即使你计划将程序员作为这类工具所产生信息的主要目标消费者，让安全团队参与进来也有很大好处。该团队能将风险管理的经验带到工作中来，并且也往往能够从整体上考虑一些安全关注的问题。但安全团队不编写代码，所以团队成员对代码的洞察力就不会有开发人员那么强。对于安全团队来说，单独进行代码的自习检查是一件艰苦的工作。事实上，即使是让安全团队设置好环境以便他们能够对代码进行编译，也是需要慎重对待的事情。如果你已经有了一套安全团队向程序员提供代码级反馈的程序，那这样做还是有帮助的。

##### 2) 程序员

程序员最了解他们的代码如何运行。将这种优势与分析工具所提供的漏洞细节相结合，因而你有很好的理由允许开发团队运行这个操作。反过来说，程序员总是处于在最后期限之前构建产品的压力之下。即使是经过培训，程序员似乎也不会有和安全团队成员同等水平的安全知识和经验。如果程序员要使用分析工具，那么要确保他们有时间能在其计划表中排出这方面的时间，并且要确保他们经过了足够的安全培训，从而在其工作中起到作用。根据我们的经验，并不是所有程序员都会成为工具的操作者。在每个团队中要指派一个高级成员来负责运行此工具，确保适当地使用结果，并回答团队内其他程序员与工具相关的问题。

#### 3) 以上全体成员

第 3 种选择就是让程序员以只产生高可信结果的模式运行这类工具，而通过安全团队来指导那些更为彻底但执行频率较低的审查。这样落在程序员肩上的负担也不会太重，同时仍然允许他们捕获某些自己的错误。这样做也促进了安全团队和开发团队之间的交流。毫无疑问，团队合作工作是最佳的。

### 2. 什么时候运行此工具?

决定什么时候运行此工具，这比其他问题更大程度地决定了组织开展安全审查的方式，就这个问题来说，有许多可能的答案，但我们最常听到的是这样 3 种：在代码编写期间、程序生成的时候以及在到达整个开发过程中的重要里程碑时。恰当的答案取决于此工具产生的分析结果将如何使用，还取决于运行此工具需要耗费多少时间。

#### 1) 在代码编写期间

众多的研究记载表明，修复一个 bug 所需的开销随着时间的推移而不断增长，因此尽快对新代码进行检查，这是有道理的。实现这一目标的一种方式是将源代码分析工具集成到程序员的开发环境中，这样，程序员就可以按需运行分析，并随时间的推移，利用此工具获得专家意见。另一种方法是将扫描集成到代码提交过程从而能集中控制这种分析。如果程序员运行此工具的次数较多，那么此工具就需要快速运行而且易于使用。对于大型项目，这可能意味着要求每个程序员只分析自己的代码部分，而后在程序生成的时候或者达到重要里程碑的时候再针对整个程序运行分析。

#### 2) 在程序生成时

对于大多数组织来说，软件项目都有一个定义完备的程序生成过程，通常都采用有规律的预定时间节点。在程序生成时执行的分析为代码审查人员提供了一份可靠的报告，此报告可用于指导问题纠正的过程，同时，此报告也可为进一步展开人工代码检查提供了一份原始资料。此外，通过将程序生成作为源代码分析的时间节点，你就创建了一种整个项目的重复、一致的估量方法，这为分析驱动的度量法提供了完美的输入。这是一种非常好的方式，通过这种方式，可以获取用于培训计划的资料。

#### 3) 到达整个开发过程中的重要里程碑时

较大程度上依赖于各种过程的组织在项目的每个里程碑处都会设置检查点，这种里程碑通常会临近开发周期结束或者是在开发期间的某些大的间隔期。这些检查点有时会包含一些安全相关的任务，比如设计审查或者渗透测试。从逻辑上对此概念进行延伸，检查点像是一个同时使用静态分析工具的天然位置。这种方式的负面影响就是，程序员可能会将对安全的考虑一直拖延到里程碑逼近的时候，在这个时候，制定别的里程碑职责，将会把安全一直推到一种边缘位置。如果你打算等到到达里程碑的时候再使用静态分析，要确保在此过程中建立一些强制性的有效手段。忽视安全的后果要能立即可见，并提前让所有人都知道。

### 3. 分析结果会如何?

当人们仔细考虑分析工具采用过程的时候，往往会忘记绝大部分的工作是在工具运行之后出现的。提前决定实际的代码审查应如何执行，是非常重要的。

#### 1) 分析工具的输出阻塞软件发布

在项目里程碑到来时，安全团队将分析工具的输出作为检查点的一部分进行处理，并对其进行优先级排序。开发团队收到排序的结果以及安全团队关于应进行修补部分的建议。然后，开发团队决定哪些问题要解决，而哪些问题归类为“可接受的风险”。安全团队应对开发团队的决定进行审查，并且，对于那些看上去开发团队所承担的比本应承担的风险要高的地方，应逐步加强用例。如果这种审查会妨碍项目按照预期实现里程碑的目标，那么软件发布就真的不那么顺畅了。如果程序员可以完全不理睬分析结果，那么他们就没有针对问题进行修改的动机。

对于维护安全而言，渗透测试是一种难以胜任的方式，同样道理，这种模型对于安全来说也是一种难以胜任的方式：它是一种反应式方法。即使软件发布并不是一种好的长期解决方案，但它可以是一种有效的达成目的的手段。希望程序员最终会厌倦了他们的发布总是被安全团队半路阻拦，因而他们决定采用一种更主动的方式来对待安全问题。

#### 2) 由中心权威机构少量发放个别结果

工具使用者的核心小组可为一个或多个项目查看所报告的问题，然后选出个别问题发送给负责问题代码的程序员。这种情况下，静态分析工具应报告所有可报告的问题，目标是不留任何死角。

因为在形成最终报告之前，熟练的分析师会处理所有结果，多疑误报的情况在这里并不是太关心的问题。通过这种模式，工具使用者的核心小组可通过少量定制熟练使用此工具，进而能够逐渐对大量结果进行熟练检查。

#### 3) 由中心权威机构设置精确的关键问题

一个组织中存在着大量项目，因而，即便审查人员的效率相当高，管理中心的效率会逐渐受限于参与结果审查的人数。不过，通常大部分严重的安全问题都会高度集中于少量类型的问题。根据这种情况，项目团队将会把工具限于针对少量特殊的问题类型而使用，这些类型的问题会根据该组织所面临的风险，随着时间的推移而发展或者变化。最后，定义一组运行良好的一定范围内的问题类型，将其作为一种集中控制的策略、标准，或者是一组指导方针。其变化的速度应该与开发团队修改并解决所有处于此范围内的问题的速度同步。从总体上来看，这种方式为人们提供了一种机会，即随着工具的使用和时间的推移，逐步积累经验慢慢成为专家。

综上所述，安全工具倾向于预先配置以检测出尽可能多的问题。如果你正试着指出某个工具能检测些什么东西的时候，这样确实挺好，不过，如果你承担着仔细检查每一个问题的任务，这种方式就不存在什么优势了。无论你如何回答关于工具采用的问题，这里我们给出的建议都是一样的：从小处着手。避开分析工具能检测的大部分东西，而后集中于小范围的重点且熟知的问题。只有当存在有使用分析工具的工作程序、且初始关注的问题已经得到控制的时候，才可以扩大检测的问题范围。无论你怎么努力，现存的大量代码都不会一夜之间变得很完美。你所在的组织内的人会因为你帮他们做出了某些优先级排序的决定而感谢你。

### 4.2.3　静态分析度量标准

源自静态分析结果的度量标准可用于排序补救工作的优先级、在多个项目之间调配资

源、从高效的安全过程获得反馈。理想情况下，应该能使用源自静态分析结果的度量标准来帮助量化与某段代码相关的风险总量，但使用工具来进行风险估量是很难做到的。最明显的问题是始终存在的误报和漏报，不过，对其进行补偿还是有可能的。通过手工方式审计足够多的结果，对于某个给定项目，安全团队可预知误报和漏报发生的几率，并从一组原始的结果中推断出正确判定的问题数量。使用静态分析来量化风险的更深层问题是没有太好的方法来计算由一组漏洞带来的风险总量。两个缓冲区溢出所带来的风险是单个缓冲区溢出的两倍吗？那么十个呢？由工具识别出的代码级漏洞完全不能算作对风险的精确刻画。

我们不应该试图使用静态分析结果来直接量化风险，而应将其作为一种战术级的方法，用以集中安全工作的焦点，并作为一种创建代码的过程的间接度量。

### 1. 战术关注点的度量

许多简单的度量标准都可通过静态分析结果实例化。我们来看看下面这些度量标准。

#### 1) 估量漏洞密度

我们已经否决了漏洞密度的度量标准，那么这里还能说点什么呢？将静态分析结果的数量除以代码行数，对于风险的估量来说，是一种糟糕的方式，但对于估量进行完整的审查所需的工作量来说，这是一种不错的方式。比较不同模块或者不同项目的漏洞密度，有助于用公式化的方法进行审查工作的优先级排序。不断跟踪漏洞密度，可以洞察工具的输出是否引起了注意。

#### 2) 通过严重性比较项目

静态分析结果可用于项目比较。图 4-1 展示了两个模块间的一种比较：

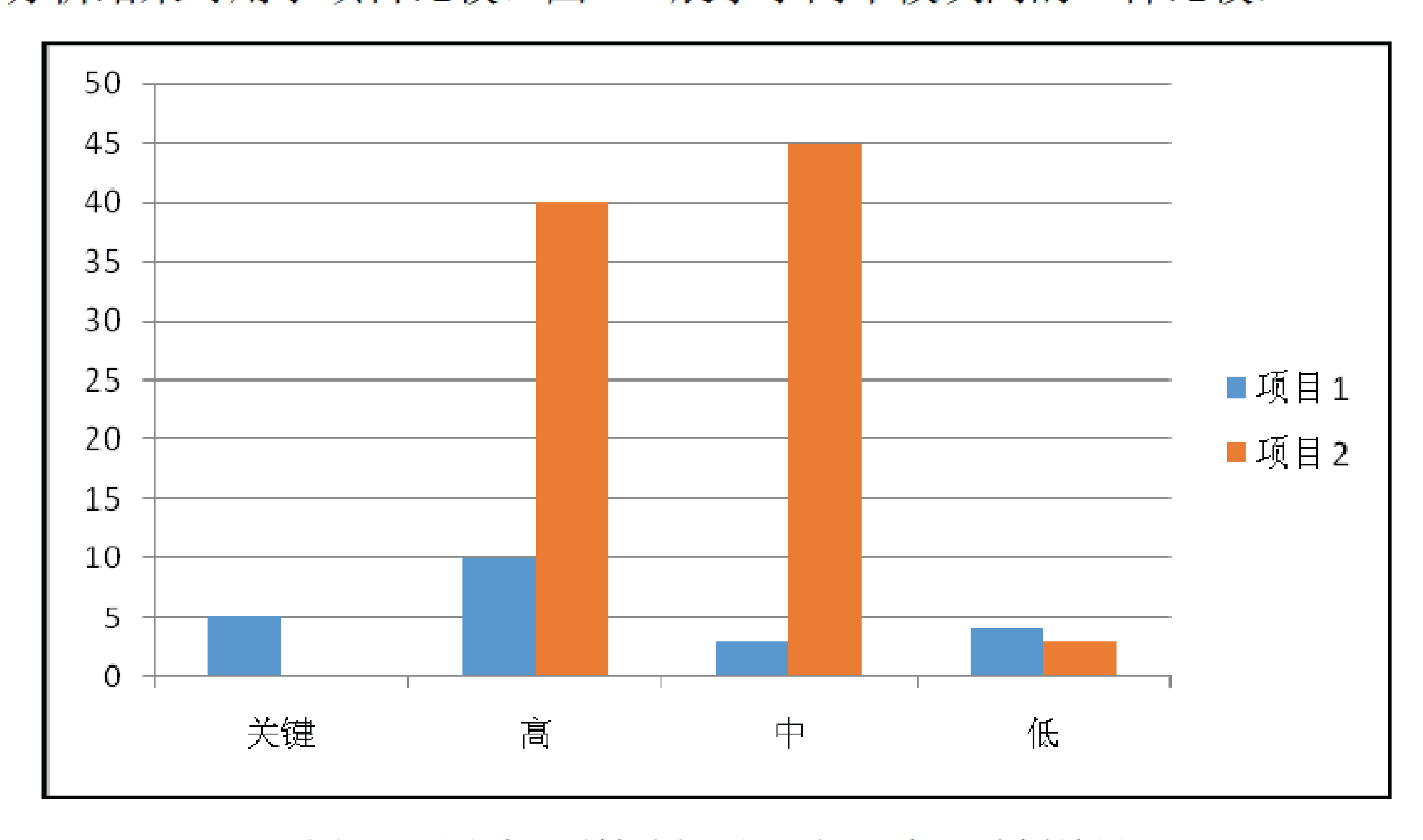



图 4-1　通过严重性分组两子项目源代码分析结果

其中就是用了通过严重性分组的源代码分析结果，这张图表提供了一种行动计划的建议：检验第一个模块的关键性问题，而后转到第二个模块的严重程度较高的问题。

通过并列比较项目，可以帮助人们了解他们所面临的工作有多少，与同事们的工作相比如何。当你提供项目对比时，要点出名字，指定到每个人。

3) 通过类别来细分结果

图 4-2 给出了通过类别划分的单个项目的分析结果。这张圆形统计图给出了解决每个类型问题大致需要的补救工作量。该图还提示我们，在即将到来的培训课程中，日志伪造和跨站点执行脚本是很好的话题。

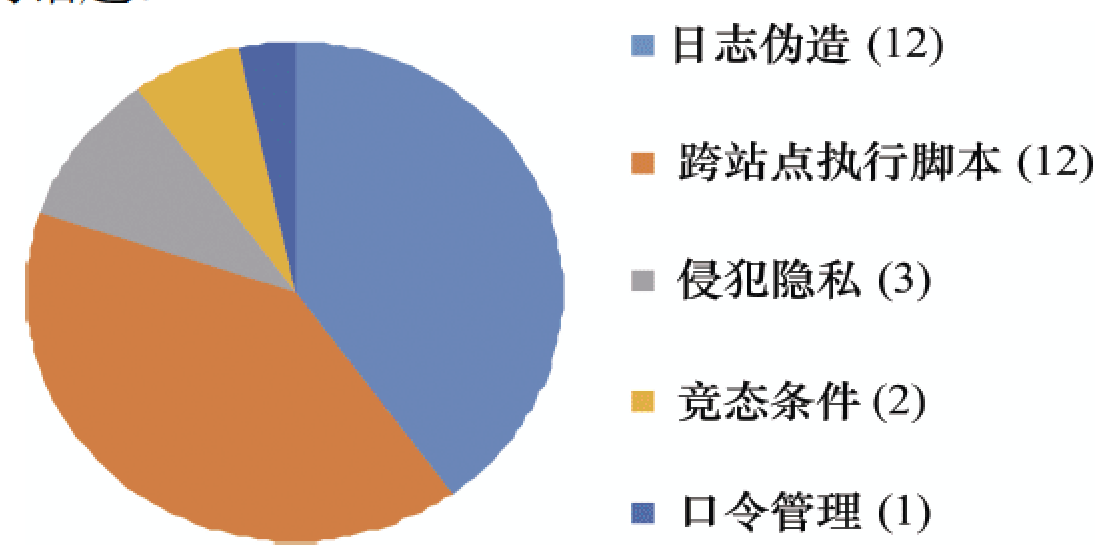



图 4.2　通过类别细分的源代码分析结果

源代码分析结果也可指出发展趋势。关注安全的团队将在他们的代码中减少静态分析的发现。发现的问题数量急剧增长的情况值得注意。

4) 对趋势进行监测

某些类型问题的绝对存在可作为一种对较普遍的安全缺陷的早期指示器。判断这类可作为前期指示器的问题，要求具有使用检查特定类别软件的某些经验。根据我们的经验，对于 C 语言编写的程序来说，大量与字符相关的缓存区溢出就是问题的一种征兆。

更复杂的度量利用源代码分析器的能力，在不同的构建版本之间针对同样问题给出同样的标示符。通过不断跟踪同一个问题，并将其与人工审计提供的反馈相结合，源代码分析器可检查项目的进展。例如，静态分析结果能揭示开发团队对安全漏洞的响应方式。在审计人员识别出一个漏洞后，程序员平均花多长时间解决这个问题？我们将这段时间称做漏洞停留期。

静态分析结果还可以帮助安全团队决定何时对某段代码进行审计。审计频率应与开发的速度保持同步。但更好的情况是，它应与潜在的安全问题引入代码的速度保持同步，同步持续跟踪某个问题的静态分析结果能向安全团队展示在一个项目中包含了多少未经审查的问题。

## 4.3　静态分析的过程

图 4-3 显示了静态分析安全工具框图。

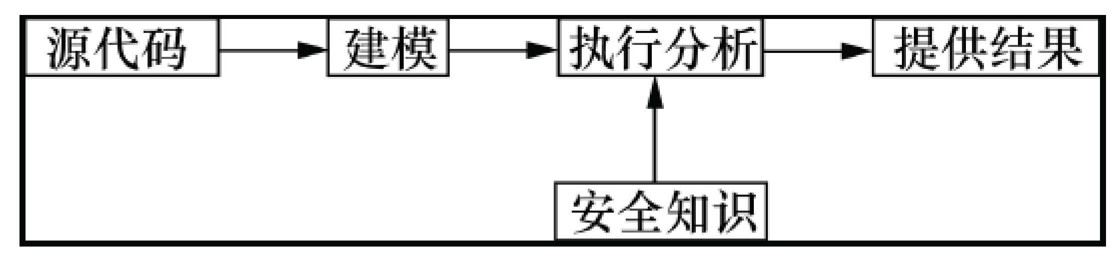



图 4-3　静态分析安全工具框图

### 4.3.1　建模

静态分析工具首先要做的，是用一个程序模型来表示待分析的代码，即提取目标代码的数据结构。该程序模型与所选择执行的分析类型密切相关，在构建该模型的过程中，静态分析工具一般会借用大量来自编译器领域的技术，其中最重要的技术如下所述。

#### 1. 词法分析

工具针对源代码进行的第一个操作，是将其转换为一系列记号，这个记号流的创建过程称为词法分析(程序文本中空白或注释等不重要的部分将直接跳过)。词法规则通常采用正则表达式来识别记号。

#### 2. 解析

语言解析器中用来匹配上述记号流的是一种与上下文环境无关的语法。该语法由一组产生式构成，用语言中的一对符号来标记。解析器通过将构成语法的产生式规则与上述记号流进行匹配而执行一种派生。如果每个符号都与派生出此符号的那个符号相连接，那么一棵解析树就形成了。

#### 3. 抽象语法

基于一棵解析树来进行重要的分析不仅可行，而且执行某些特定类型的格式检查会收到很好的效果。这种树中的节点是直接从语法的产生式规则派生而来的，而这些规则会引入一些非终结符号，这种符号的存在纯粹是为了使得解析易于进行而没有歧义，而不是为了产生更易理解的树；通常来说，要进行这种分析，较好的做法是在程序文本中，提取语法中的细节和句法中的修饰成分。完成这项工作的数据结构称为抽象语法树(AST，Abstract Syntax Tree)。AST 一般是通过将语法的产生式规则与树状代码结构相关联而构建的，其目的就是提供一种标准程序版本以供后期分析之用。

根据系统的需要，AST 可包含一个比源语言数量更有限的结构。例如，对方法的调用可能会转换为对函数的调用，或者 for 和 do 循环可能转换为 while 循环。以这种方式对程序进行重大简化称为降级。紧密相关的语言，如 C 和 C++，可被降级为同一种 AST 格式，不过，这种降级也带来了歪曲程序员意图的风险。句法上相似的语言，如 C++和 Java，可共享许多相同的 AST 节点类型，但毫无疑问会有一些特殊类型的节点仅用于各自的语言中。

#### 4. 语义分析

分析工具在生成 AST 的同时，也会生成一张符号表。该符号表将程序中每个标识符的类型和声明或定义的指针与其相关联。在得到 AST 和相应的符号表后，分析工具执行类型检查所需的准备工作宣告结束。

在编译器领域，类型检查和符号分析都是作为语义分析的概念而提及的，这是因为编译器会给程序中发现的这些符号赋予一定的意义。使用这些数据结构的静态分析工具比起那些没有使用的工具来说，具有明显的优势。

在语义分析后，编译器和较高级的静态分析工具就将分道扬镳了。现代编译器利用 AST、符号和类型信息生成一种中间的表示法(便于优化的通用版本机器码)，之后转换为平台特有

的目标代码。静态分析的过程就比较复杂了，静态分析工具根据所执行分析类型的不同，可能需要以 AST 为基础执行额外转换，或者生成特有的中间表示法变种。

如果静态分析工具使用自己的中间表示法，那么通常至少会允许赋值、分支、循环和函数调用。静态分析工具所使用的中间表示法通常是比编译器使用的中间表示法更高级的程序视图。例如，C 语言编译器可能会为自己的中间表示法而将所有对代码字段的引用变换为此结构内的字节偏移，而静态分析工具则更可能继续通过其名称来指向结构字段。

### 5. 跟踪控制流

许多静态分析算法都会探究函数执行可能采取的路径。因此，绝大多数分析工具都会在 AST 或者中间表示法之上生成一个控制流图以使算法更高效。控制流图中的节点是一些基本块：指令序列总以连续方式从第一条执行到最后一条。另外，其边界执行代表着基本块之间隐含的控制流路径，返回边界代表着隐含的循环。

程序运行时的控制流可用其执行的一系列基本块来描述。控制流的轨迹是一个基本块组成的序列，这个序列就是一条完整的代码执行路径。

调用图描述了函数或者方法之间潜在的控制流。如果缺少函数指针或者虚函数，我们仅是通过查找每个函数中引用的函数指针来构建调用图。在构建好的调用图中，节点表示函数，有向边缘表示隐含的调用。

在函数指针或者虚函数被调用时，分析工具会将数据流分析和数据类型分析组合使用来限制从某个调用点上可能引用的函数集合。在运行时动态载入代码模块的情况下，因为那些在分析时不可见的代码可能会被程序运行，所以此时没有能确保控制流图完整无缺的方法。

对于交叉使用多种编程语言的软件系统，或者由多个相互协作的程序构成的软件系统，静态分析工具将完美地缝合出一张表示各部分之间连接的控制流图。对于某些系统来说，配置文件中保存着调用图在不同环境中所需的数据。

### 6. 跟踪数据流

检查数据在程序中的移动的路径通常采用数据流分析算法。编译器执行数据流分析，从而定位寄存器、移除无用代码、并执行许多其他的优化。

数据流分析一般是对某个函数的控制流图进行遍历，同时记录数据值的产生和使用位置。将函数变换为静态单一赋值(SSA，Static Single Assignment)形式，这对许多数据流问题来说都是有益的。SSA 形式的函数只允许对每个变量进行一次赋值。为适应这种约束，就必须将新变量引入到程序中。

如果某个变量分别沿着不同控制流图被赋予了不同的值，那么在 SSA 形式中，这个变量必须在控制流路径融合的那个位置进行合并处理。SSA 引入一个新型变量并将来自两条控制流路径之一的值赋给这个新型变量，通过这种方法来完成上述融合。用于融合点的这种简化符号称为一个 ψ-函数。

### 7. 污染传播

安全工具需要知道程序中的哪个值可能会被攻击者控制。污染传播通过数据流来判定攻击者能够控制的值，它要求知道数据在程序中的进入地点以及传送方式。污染传播在对许多输入验证缺陷以及表示缺陷的识别中起着重要的作用。例如，一个程序中如果含有一个可被

利用的缓冲区溢出漏洞，那么一般也会包含一条从输入函数至有漏洞操作的数据流路径。

#### 8. 指针别名歧义

数据流的另一个有关问题是指针别名分析。其目的是获知可能会指向相同内存地址的那些指针。别名分析算法采用“必需的别名”，“允许的别名”和“不允许的别名”这样的术语来描述指针间的关系。许多编译器优化需要某种形式的别名分析以确保正确性。

静态分析工具假定指针(至少是作为函数传递的指针)不存在别名，这种情况是比较常见的。对于大多数工具来说，进行这种假定往往看起来足以产生有用的结果，但这样会使工具遗漏掉重要结果。

### 4.3.2 分析算法

提高上下文环境敏感度并逐一判断特定代码的运行环境及运行条件，这就是使用高级静态分析算法的动机。随着上下文环境敏感度的提高，我们就能更好地评估目标代码可能带来的危险。要指出所有调用 strcpy()的地方并将其替换是比较容易的，难的是仅关注那些可能为攻击者开展缓冲区溢出攻击提供了可能性的 strcpy()调用。

所有高级分析策略的构成都至少包含 2 个主要组件：一个是程序内分析组件，用来分析单个函数；另一个是程序间分析组件，用于分析函数间的交互行为。由于程序内和程序间的名字很相似，所以我们使用更通俗的术语“本地分析”来表示内分析，而用“全局分析”来表示程序间分析。

#### 1. 断言检查

断言可用来描述许多安全属性，而且，这些断言对要保护的程序来说必须是真实的。

如果程序的逻辑保证这个断言总是有效的，就不可能发生缓冲区溢出。如果存在一组条件，在这些条件下这个断言可能失效，那么分析程序就应该报告一个潜在的缓冲区溢出错误。利用这种断言的方法同样能很好地进行需求定义，这些需求可用于避免出现 SQL 注入、跨站点执行脚本等。

通常有以下 3 种从安全特性产生的断言：

1) 大部分安全问题都由程序员引起，他们在不该信任的时候信任了输入，因此，分析工具应该检查那些与在程序中传送时数据被赋予的信任等级相关的断言。这些都属于污染传播问题。SQL 注入和跨站点执行脚本是两种漏洞类型，它们将使分析工具构造有关污染传播的断言。在最简单的情景中，数据的值或者被污染了或者没有。另一种情况是，仅一段数据就可能携带了一个或多个特定类型的污染。

2) 查找可利用的缓冲区溢出漏洞所引发的断言与源自污染传播的断言相似，但判断一个缓冲区是否发生溢出，它要求跟踪的远不止是否涉及污染的数据；分析工具还需要知道这个缓冲区的大小以及用作索引的那个值。由于它要求知道一个变量可能有的潜在值的范围，所以也将这些分析称为范围分析问题。

3) 某些情况下，工具不太关心具体的数据值，而更多关心某个对象在程序运行时的状态。这称为是类型状态——在代码中的每个位置，变量都可以有不同的类型。

### 2. 本地分析方法

抽象解释：是一种通用技术，这种方法首先将程序中与所关注的属性无关的信息抽取出去，而后使用选中的程序抽象执行一种解释。

由循环带来的问题可通过执行一种流敏感分析来加以解决，这种分析没有考虑所执行语句的顺序。从程序员角度看，这可能看起来是一种相当极端的方法；语句在程序中呈现的顺序是非常重要的。不过，流敏感的分析保证所有语句排序方法都经过考虑，这为考虑到所有可行的语句执行顺序提供了保障。这种方法排除了对复杂的控制流分析的需要，但其代价是许多不可能的语句执行顺序可能也会进行分析。那些致力于控制误报的工具通常至少会部分尝试使用流敏感的分析，以便不会报告那些当程序运行时几乎永远不会发生的语句顺序所带来的问题。

谓词转换器：一种用来替换模拟和解释的方法是导出函数对其调用者的需求。Dijkstra引入了一种最弱前置条件的概念，这个概念源自一个使用谓词转换器的程序。程序的最弱前置条件是对程序调用者的一组最少的需求，这些需求是实现最终状态所必需的。程序中的语句可以视为是对后置条件执行转换。从程序中的最后一条语句开始倒退处理到第一条语句，程序所需的后置条件被转换为该程序要成功实现所需的前置需求。

模型检查：对于临时性的安全属性，比如“内存应该只释放一次”以及“应该只有非空指针才能被解除引用”，可以很容易地将所检查的属性表示为一个小型的有限状态自动机。模型检查方法将此属性当成一种规范，同时将待检查程序转换为一种模型，此后对比上述规范与模型。

### 3. 全局分析

忽略问题是最简单的全局分析方法，另外，如果一次检查程序中的一个函数，则假设所有存在的问题都会自我证明。这对许多安全问题来说显然是个错误假设，而对于那些与表示法和输入验证有关的问题，由于识别它们常常是要求跨函数边界来检查的，这种假设更不可取。

整个程序分析则是最复杂的全局分析方法，其目的是以完全了解程序调用函数上下文环境为基础，对各个函数进行分析。另外，从概念上讲，直接插入也算一种进行整个程序分析的简单方法，该方法以被调用函数的定义代替程序中的各个函数的调用。另一种方法也是一种可能的办法，这种方法涉及使用一种基于堆栈的分析模型。不管采用什么技术，整个程序分析将要求大量的时间开销和/或内存开销。

利用本地分析算法来创建函数摘要则是一种更灵活的全局分析方法。利用该方法，当本地分析算法遇见一个函数调用时，将把函数摘要作为函数的替代来使用。函数的摘要可能会极其精确或者极其模糊，这就使得摘要生成和存储算法在精确度和可扩展性之间进行折中变得可行。函数摘要可能既包含调用上下文环境必须满足的需求，又包含该函数返回时对调用上下文环境所带来的影响。

## 4.3.3　规则

就算不比分析算法和工具所实现的启发式方法更重要，定义了安全工具应该报告些什么的规则起码也和前两者同等重要。分析算法有时还可能由错误的“因”获得正确的“果”，

但分析工具永远也不会报告超出规则集范围的问题。

早期的安全工具有时会通过计算每个工具默认情况下打包进来的规则数量，对工具进行简单比较。最近的静态分析工具就更难比较了。多个规则可能会一起起作用检测一个问题，而单条规则可能会指出抽象接口或者根据正则表达式来匹配方法名称。就像代码更多并不总是能带来更好的程序一样，更多数量的规则并不总是带来更好的静态分析工具。

代码质量工具有时会从其分析的代码来推断规则。如果某个程序在 100 个不同的位置调用同一个方法，而在其中 99 个位置该程序都注意了该方法的返回值，但仍会有相当大的机会在那个剩下的位置中没有检查返回值。因此，这种统计方法来推断规则并不能很好地适用于对安全问题的识别。如果程序员不了解某种具体构造所代表的安全风险，那么其代码可能就会在整个程序中一律使用这种错误构造，如果只给出统计方法的话，这种情况会造成 100%的漏报率。

规则不仅用于定义安全属性，也会定义那些在程序文本中未明确包含的程序行为，如该程序使用的任何系统或第三方库的行为。为系统库和流行的第三方库创建并维护一个优秀的建模规则集合是一项巨大的工作。

### 1. 规则格式

在程序外部实现检查代码所需的规则是每一个好的静态分析工具都会尽力实现的功能，因为如果实现了该功能，就能在无须更改工具本身的情况下修改、添加或者减少规则。最好的静态分析工具会将所有要检查的规则都放在外部实现。除了调整外部工具的行为外，外部规则接口还需让最终让用户添加检查规则来适应新类型，或者以分析程序语义的方式扩展现有的检查规则。

专用的规则文件：在维护一份外部文件时，在其中使用一种专门格式的描述规则；这样做将允许针对分析引擎的能力来定制规则的格式。

批注：有时，以批注形式将规则附在程序文本中会收到更好的效果。如果某些专门的规则决定着某个具体模块的使用，将这些规则直接放在此模块中(或放在此模块的头文件中)是一种不错的方式，通过这种方式，可确保在此模块使用的时候应用这些规则。批注的上下文环境从其周围的代码便可看出，因此，它常比以外部文件形式存在的规则更简短。例如，批注无须详细说明某个函数的名字，只要在此函数声明之前出现即可。

此外，这种批注形式也存在一定的劣势。例如，若分析人员并不是代码的维护者或者所有者，则很可能不会允许他们将永久性的批注添加到程序中。而通过创建几乎只包含批注的专用源文件并将这些源文件只用于分析适用的方式，分析者或许能够克服这类限制。

### 2. 用于污染传播的规则

使用静态分析解决污染传播问题需要各种不同的规则类型。由于很多安全问题可被表示为污染传播问题，故这里简要概括一下各种污染传播规则的类型：

- 源规则定义了受污染的数据进入系统的程序位置。
- 接收器规则定义了不应该接受受污染数据的程序位置。
- 传递规则定义了函数操纵受污染数据的方式。
- 净化规则是传递规则的一种形式，用于从一个变量中移除污染。净化规则用于表示

输入验证函数。

- 入口点规则与源规则类似，在这里，将污染引入到程序中，但入口点规则不是在程序中函数调用的位置引入污染，而是由攻击者调用函数的时候引入污染。

在最简单的形式中，污染是一段数据的一个双态属性，其值要么是已污染，要么是未污染。实际上，输入法验证问题并非这样泾渭分明。输入会由于某些意图而受到信任，也会由于其他原因而不被信任。为表示这样一种事实，可将各种受污染的数据作为各种污染的载体来建模。污染标记可用各种方式来应用。

首先，不同的源规则会引入带有不同污染标记的数据。来自网络的数据可能被标为 FROM_NETWORK，而来自一个配置文件的数据可能被标为 FROM_CONFIGURATION。如果这些污染标记能带入到静态分析的输出中，那么这些信息将允许审计人员根据非受信数据来源而对输出进行优先级排序。

其次，接收器函数只有在带有某种污染的数据到达时才是危险的。只有当接收到任意可被用户控制的数据时，跨站点脚本执行接收器才是存在漏洞的，而如果只收到数值数据的话，就不会有漏洞。

源规则、接收器规则以及传递规则会以一种追加或者删减的方式来操纵污染。在采用删减方式的情况下，源规则引入的数据可能携带所有需要关注的污染标记。给定其所执行的验证类型，采用传递规则建模的输入验证函数会去除适当的污染标记。接收器规则会基于受污染数据来检查危险的操作，或者，如果仍然存在那些污染标记的话，接收器规则还会检查那些逃脱应用层或者触发器检查的受污染数据。在采用追加方式的情况下，源规则以一般的方式引入受污染的数据，而输入验证函数基于它们所执行的验证类型来追加污染标记，验证跨站点脚本执行攻击的函数会追加一个 VALIDATED_XSS 标记。接收器规则扮演着和采用减去方式情况时一样的角色，当在一个不带有适当污染标记的自变量参数上执行危险操作时，或者在数据没有必要的污染标记的情况下离开应用层的时候，会触发这类规则。

### 4.3.4　报告结果

大多数学术研究工作所投资的静态分析工具，都将精力花在了创造新方法来识别缺陷方面。但是，当某个工具真正面临着投入实际使用的时候，该工具报告结果的方式对于此工具所提供的价值具有重要的影响。静态分析工具的用户倾向于使用术语“误报”来指那些可能出现在标题“非期望的结果”之下的东西。尽管这并非我们所使用的定义，但我们当然理解这种倾向。从用户角度看，并不关心底层分析算法的品质多么优良。如果你不能理解这个工具告诉了你什么，这个结果就是没有用的。从这种意义上讲，从糟糕的表示法中可以很容易地产生出糟糕的结果，就如同从分析错误中产生出的结果一样糟糕。

以用户能预测结果潜在影响的方式提供结果，这是分析工具的部分任务。像跳转到定义这样的简单代码导航功能也是很重要的。如果一个静态分析工具可作为程序员集成开发环境中的插件来运行的话，对每个人都有好处：程序员熟悉代码导航设置，而静态分析工具开发人员不必彻底重做代码浏览功能。

审计人员如果想要管理工具的输出，那么要求该工具至少具备 3 种功能：

1) 将所报告的结果进行分类并分组；

2) 消除所报告的结果中非预期的部分；

3) 对所报告结果的意义进行解释。

### 1. 将所报告的结果进行分类并分组

如果用户能以一种灵活的方式对所报告的结果进行分类并分组，那么，所报告结果中的大量非预期成分通常能够被消除，因而不必再对每一个问题逐一进行审查。例如，如果所分析的程序从一个可信的文件中获取某些输入，那么，用户将可通过消除那些假定此文件不可信的条件下所产生的所有结果，从而从对结果的审查中获益匪浅。

由于静态分析工具会生成大量结果，而如果这些结果以一种分级次序提供，使得最重要的结果最可能出现在审查的初期，将更有益于用户的分析。静态分析工具有两种将结果分级的维度。假定此工具不会犯错，那么严重性将给出所发现问题的重要性。例如，缓冲区溢出通常就是一种比释放空指针应用更严重的安全问题。可信度给出了一种对所找出问题的可能性进行评价的标准。通常情况下，为实现某个结果，工具必须要做的假定越多，那么这个结果的可信度就越低。要创建一种分级，分析工具必须对每个结果组合使用严重性和可信度得分来进行判定。典型情况下，严重性和可信度可进一步压缩为一种关于重要性的简单的非连续范围，如关键(C)、高(H)、中(M)以及低(L)。这为审计人员提供了对其工作进行优先级排序的一种简便方法。

### 2. 消除所报告的结果中非预期的部分

对非预期的结果进行审查令人不感兴趣，而对同一个非预期的结果进行多次审查简直让人疲惫不堪。对输出结果进行抑制是任何一个高级静态分析工具必备的机制，只有这样，已经报告过的问题在后续分析中将不会重复进行报告。如果这个系统比较好的话，抑制信息将会带到将来由同一代码库生成的版本中。同样，审计人员也应该能够共享和合并抑制信息，这样多个人就不需要对同一个问题进行审计。

用户应该能够关闭整个属于警告的类别，但他们仍需逐一消除错误。许多工具都允许使用编译指示或者代码批注来抑制结果。但是，如果执行代码审查的人没有修改代码的权限，那么就需要有一种抑制信息在代码之外保存的方式。一种可能的方法就是仅保存代码文件名、问题所在的行号以及问题类型。这种方法的问题是，即使是对文件的一次小小的改动，也会引起所有行号的偏移，从而会使所保存的这种抑制信息失效。通过将行号信息保存为从所在函数的开始位置的偏移量，或者是从最近标记的语句开始的偏移量，可以减轻这种方法存在的问题。如果结果中包含程序中跟踪的信息而不仅是一个行号，那么第二种方法就尤其有用。这种方法基于组成结果跟踪轨迹的程序构造为结果生成一个标识符，其中包括函数和变量的名称、相关的部分控制流图以及与确定结果有关的所有规则的标识符。

### 3. 对所报告结果的意义进行解释

人工测试出来的优秀 bug 报告应包括问题描述、关于此问题会影响到谁、解释为什么它重要以及要再现此问题所需的步骤。但就算是优秀的 bug 报告有时也会送回标有“不能再现”或“不是问题”的内容。当出现这种情况时，测试人员会再次尝试解释这种情况。静态分析工具不会进行再次尝试，因此他们必须在第一次的时候就做出一种有效的论证。由于程序员可能不能立即理解所发现问题的安全衍生意义。这种情况下，发现出来的这种问题可能难以

吸引未经培训者的眼球，但实际上却会很容易被攻击者挖掘出来。因此，静态分析工具必须能够对其所报告结果的意义进行解释并对潜在的安全隐患进行预测。

# 4.4　静态分析中的常见问题

## 4.4.1　处理输入

开发人员所能采用的最重要防范措施是验证应用程序所接收的全部输入。输入验证和表示方法是最重要的，因为不检测输入或者不能够正确地检测输入会导致某些恶意攻击，包括缓冲区溢出、SQL 注入攻击以及其他许多攻击。

程序员最有资格确定在他们的应用程序中哪种输入才是合法有效的。这种判断在需要依赖正确输入来产生正确结果的情况下显得更加重要。程序员可能不能够确定所接收到的所有输入是否都是正确的，却可以保证它们不具有明显的错误。不要期望所有输入都具有正确的格式，更不要因为某个输入的来源似乎是可靠的或者安全的，就完全相信它。不要因为某个输入是由你所编写的程序代码来生成的就相信它；你的应用程序也许是从某个不太可信的来源接收到这个输入，或者这个可信来源本身就受到安全威胁。当输入验证程序确定输入存在问题时，应当拒绝接受输入，不要尝试纠正问题然后继续接受输入。简而言之，怀疑所处理的一切输入，无论如何，保证应用程序的安全是首要任务。

程序员不得不接收用户的输入，但又不能够相信输入，因此严格检测输入并验证其正确性，将输入的值限定在某个能够接受的范围内才是明智的做法。这个过程，通常称为输入验证。

### 1. 验证内容

许多应用程序接收多个来自于不同来源的数据，并利用这些数据进行各种运算操作。无论最初是从应用程序的外部读取数据，还是在具体安全环境中使用这些数据，输入验证都在软件安全性中扮演了重要角色。本节讨论输入验证的两个方面：需要验证的输入的种类、依赖于验证后输入的操作的种类。

#### 1) 验证所有输入

本节的核心思想是：验证所有输入数据。仔细思考这个问题，我们不要认为只有用户从输入界面输入的数据才是输入数据，而要将输入数据的范围定义得更广泛一些。另外，如果一个应用程序包括多个过程，则每个过程的输入数据都需要进行验证，即使该输入数据来自于同一程序的其他过程也同样如此。再次，即使输入数据是通过一个安全连接传递过来的，或者访问应用程序的用户只能是可信用户，也要对其进行验证。最后，来自操作系统环境的输入数据也不能轻易相信，这其中当然包括路径名称和注册表值。应用程序执行的每次验证都拒绝了攻击方的一次攻击机会，并增加了自己的安全系数。

如果已经确定应用程序的逻辑和功能，则可将输入验证程序分为两个主要部分：语法检查和语义检查。前者主要检测输入格式正确与否，它通常可以远离应用程序的逻辑部分，紧

接着数据输入模块执行；后者主要确定输入恰当与否，由于与逻辑密切相关，它通常紧靠着应用程序的逻辑部分。

在编写应用程序时，要达到如下目的：如果这个应用程序将来由其他程序员来维护，在不扩展验证逻辑以覆盖新输入点的情况下，后来的程序员无法或者很难在应用程序中添加一个新输入点。输入验证逻辑程序应该要求应用程序将所有的输入数据都传递进来并且拒绝所有未通过验证的输入。

#### 2) 验证各种来源的输入

不要将软件的安全寄托在配置、使用和维护人员的敏锐理解力、深刻洞察力或良好意愿上。安全不仅需要验证用户输入，还需要验证所有来自软件之外的输入。这些输入应当包括下列内容，但并非局限于此：命令行参数、配置文件、环境变量、注册表值、数据库查询、临时文件。

常会有一些开发人员不愿对各种来源的输入都进行验证，他们的理由是："我并不认为会有人从这些地方来攻击我的程序，为什么我必须要耗费这个多时间和精力来编写这些输入验证程序，以保证软件的安全呢？"这种态度显然会导致软件的一些盲点，使得软件易于受到攻击。目光短浅的程序员也许会想："如果攻击者已经具备修改系统性能参数的能力，就已经达到目的了。"实际上，攻击者可以很轻易地找到方法略微改变一下系统文件或者某个脚本，或者也可以伪造出一个真实的配置错误。无论哪一种情况，缺乏输入验证都是成为攻击者攻破整个系统的途径。并非所有的输入都是相同的。对于来自配置文件的输入所进行的验证往往不同于对用户输入所进行的验证。但无论来源如何，所有输入都至少应当进行一致性验证和语法验证。

### 2. 验证方法

一旦确定在应用程序中的哪些地方需要进行输入验证，就必须确定选择哪种策略来完成验证。

- 使用强输入验证

进行输入验证的最好方法就是根据一系列已知的正确值来检验输入。正确值输入验证法并不尝试检验某些特定的错误值。这就类似于一个需要邀请函才能参加的晚会。如果需要保证只有收到邀请的客人才能入场，最好的方法就是开列一个客人清单，并且在入门时核对。千万不要试图列出一个没有被邀请人员的清单来进行校验。

通过已知正确值的清单来进行输入检验称为白名单法。当可能输入的集合比较小时，可以使用间接选择来确保不能绕过白名单。当进行输入验证时，间接选择是首选方法。最差的方法是黑名单方法：尝试枚举所有的不能接受的输入值。

1) 间接选择：一般来说，应用程序所能容忍的意外输入的次数越少，其安全性也会越高。因此，利用一个间接输入层进行输入验证应该算是最好方式：列出一个包含所有合法有效数据的清单，并且只允许用户提供包含在该清单中的特定索引值。通过这种方法，应用程序逻辑就不需要直接面对用户提供的输入，而只需验证用户是否提供了一个合法的索引。虽然这种输入验证方式有点脱离实际，但当用户需要从很多值中做出选择时，这是最理想的方式。利用值选择法可以保护数据。可消除任何直接使用输入数据的企图。

2) 白名单法：由于合法值范围太过广泛，所以间接法在许多情况下是不可行的。这种情况下，最好创建一个合法输入值的白名单，并通过其中的选项组合生成有效输入。与间接法不同，输入值可以任意方式组合而成，从而使有效输入的范围大大扩大，而不仅限于一个预定的范围，例如，可通过数字和标点符号的集合来组成任意电话号码。间接法和白名单法以同样的处理方式来对待意外输入：拒绝接收。这两种方法都只接收确认合法有效的输入。如果需要应用程序接收某个新的值或者字符，必须要重新更改这些验证逻辑程序以便接收新值。否则，这些值只会被拒绝。

- 避免黑名单法

黑名单法是当采用白名单法无法轻松实现时的第二选择。它有选择性地拒绝或者逃避潜在的危险输入值或者序列，也就是说黑名单法只拒绝那些已知恶意的数据。由于在给定环境中，恶意值的集合一般难以枚举，甚至可能是无限的，这就使得黑名单法是不完善的。总之，黑名单法是不利于应用程序安全的。

黑名单法具有明显的危害性，因为它也许可以很好地防范某些基本的攻击行为，例如阻止攻击者通过 HTML 文件中的<script>标记来进行攻击，从而使得开发人员陷入一种虚假的安全感中，但当进行各种试探性攻击时，这种方法往往起不到防范作用。如果程序员用一个简单的攻击实例来测试一个易受攻击的 Web 站点，当攻击失败时，他会误认为这个站点是安全的，但实际上，并不是这样。攻击者所需要做的事情就是，确定哪些字符没有列于黑名单中，或者使用其他适当的编码方式来完全绕过黑名单验证。

实际上，当输入经过验证后又要进行变换时，黑名单法通常都是失败的。通过解压缩、解码、改变字符集、剥离多层编码或者通配符扩展等操作，都可以改变输入的含义。

- 不要混淆可用性和安全性

不要将应用程序为了可用性目的而执行的验证，和为了安全目的所进行的输入验证混为一谈。出于用户界面友好目的而进行的输入验证，通常是为了在合法用户犯错时捕获常见错误，提供易于理解的反馈。为安全目的进行的输入验证是为了处理非同寻常的、不友好的输入。

- 拒绝不良数据

有些程序员试图修复某些已经验证失败的输入类型。如果用户遗漏了某个必须输入的字段，可以设置一个默认值；超出最大长度限制的口令字段可以截断；输入字段值的 JavaScript 程序可代之以转义字符。不要试图修复未通过输入验证的输入。相反，要坚定不移地拒绝接收这种输入。

输入验证本身就是一个复杂的主题，但在程序中既有执行输入验证的代码，又掺杂着自动纠正输入错误的代码，当数据以多种方式编码或者可以分层或者嵌套编码时尤其如此，那程序的复杂性就会呈爆炸性地增长。用于自动纠错的程序可能会改变输入数据的本意，或者使得用于验证输入的逻辑程序受到影响。

如果攻击者想要输入列于黑名单的某个单词，怎么办呢？当然，选择另一个没有出现在黑名单中的单词是一个不错的选择，但如果不使用某个列于黑名单中的词就不能够正确表达意图，又该怎么办呢？比较简单的方法是，将一个列于黑名单中的词插入到需要输入的单词中。例如，这个纠错程序正好可以将输入“bastpruntich”纠正为“bastich”。在这种自动纠错的应用程序中，不能正确处理这种多层编码的输入是常见错误，而且可能导致严重后果。

不能为攻击者将某个输入转换成另一个输入。直接拒绝没有通过输入验证的数据。当考

虑是否需要在输入验证程序中加入数据纠正功能时，先问问自己，谁可能提供这些不良输入。从安全角度看，存在两种可能：偶然失误的用户或者故意试图破坏规则的攻击者。对于前者，合法用户可能比较赞成应用程序指出错误输入以引起他们的注意，并且允许他们纠正自己的错误，以免将来再犯同样的错误。对于攻击者，需要记住，无需为试图破坏应用程序正常运行的输入数据忙碌。

● 默认执行正确的输入验证

无论使用何种编程语言，用于接收输入的标准方法都没有提供一个内置输入验证功能。这意味着，每当程序员需要添加新的代码来检索某种新输入时，都需要重新处理整个输入验证过程。

不要每次都为输入验证问题重新编写一个新的解决方法，而需要在应用程序中清晰、一致地编写输入验证函数。这意味着，需要在应用程序用于获取输入的系统类库的顶层编写一个抽象层。通过自己编写函数或者方法层来取代内置函数，可使用程序默认执行正确的输入验证。我们将此称为强安全 API。

当程序员使用安全 API 时，相比直接调用内置方法而言，通常需要提供更多的参数。这些附加参数将携带用于在调用函数中执行正确的输入验证所需的信息。

我们不准备介绍这样的输入数据过滤程序，可将同样的准则用于应用程序所接收的任何输入。绝大多数情况下，只在应用程序前端提供的环境下，很难实现正确的语法和语义检查。

强安全 API 提高了以下几个方面的能力：

1) 对所有的输入提供与上下文相关的输入验证。相比每个模块都单独实现自己的输入验证来说，采用这种方法，困难在于建立一致的输入验证策略。

2) 理解并维护输入验证逻辑。输入验证是不容易处理的。多种不同的实现方式使得程序需要执行的测试与确认工作量大大增加，使得一项本来就很艰巨的任务变得更加困难。

3) 可连续一致地更新和修改输入验证方法。当发现输入验证逻辑中的错误时，能否修正这些错误？如果没有集中实现逻辑验证程序，恐怕答案只能是“我不知道”。

4) 始终如一。如果不默认执行输入验证，程序员很可能会忘记编写这些程序。通过将输入验证作为标准处理过程的一部分，可保证应用程序接收到的所有输入都是合法有效的。

不要过度重视通用性也很重要：避免降低输入验证的严格性来换取更多的通用性。只有在验证逻辑程序功能强大到足以增强输入所需的限制，以保证其在所使用的上下文环境中足够安全时，才能重复使用验证逻辑程序。

● 检验输入的长度

前面的验证逻辑通常检查输入的长度是否位于一个最小值和最大值之间。长度检查通常比较容易做到，因为并不需要了解输入的含义就可以实现。但需要注意的是，如果应用程序在处理输入以前先进行输入转换，则输入还可能会变得过长。

输入验证的最基本功能是执行长度检查，这是其他功能的前提。在输入验证期间，最好能够尽可能多地考虑应用程序的上下文环境。在应用程序需要验证某个输入字段时，验证逻辑如果尽可能多地了解到这个输入字段的合法值，就可以更加严格地执行验证。

良好的程序设计习惯是，千万不要将前端验证代码和业务逻辑紧密地交叉混合在一起。这样做的后果就是，验证程序很少能够具备完善的上下文关系来最好地验证输入。依赖于应用程序的上下文环境关系来区分前端输入验证和混杂在应用逻辑中的验证检查；但在前端验

证中至少要包含输入长度验证。

验证输入是否超过长度上限，可以使得攻击者难以利用系统的其他漏洞。例如，如果某个输入字段可用作跨站点脚本攻击的一个部分，与只能输入少量字符的攻击者相比，可以编写任意长度代码的攻击者具备更大的灵活性。通过检验输入的长度是否小于最小输入长度，攻击者不仅无法忽略必须输入数据的输入字段，而且不能因为输入数据长度短而导致数据有效。

- 限制数值输入

检查数值输入是否介于最大值和最小值之间也是输入验证的一部分。应当警惕某些操作可能传递超过最大值和最小值范围的数据。

当攻击者利用整数变量的有限容量时，我们称之为整数溢出。在 C 和 C++中，比较典型的方式是，将整数溢出作为缓冲区溢出攻击计划的一个组成部分。Java 中的整数溢出并不会导致缓冲区溢出攻击，但仍会导致其他不良后果。

避免整数溢出问题的最好方法是，检验所有的整数输入是否都位于上下界范围内。理想情况下，上下界的选择应当使得任何后续计算结果都不会超出所使用变量的容量限制。如果这样的界定范围过于严格，则应用程序必须包括内部检验以保证所计算的数值不会导致溢出。

### 3. 防止元字符攻击

允许攻击者控制发送到数据库、文件系统、浏览器或者其他子系统的命令，会导致很大的灾难。绝大多数程序员都没有愚蠢到故意给攻击者提供直接访问这些子系统的权限，但是因为接口设计的方式问题，很可能无意中就提供了这种便利。

所有强调易于使用或者提供交互能力的脚本语言和标记语言至少有一点是相同的，它们可以接收将控制结构和数据随意结合在一起的字符串。

问题在于，如果程序员不加特别注意，则可能会提供添加、删除或者改变控制结构含义的能力，尽管程序员的本意只是允许用户列举某些特定数据，这是很不明智的。攻击者常通过指定元字符来利用这个漏洞实施攻击，所谓元字符就是在输入语言中具有特定含义的字符或者字符串。事实上，通常对于同一个元字符存在多种编码方式；而且同一语言的不同编程方式，可能会接受不同元字符或者元字符的不同编码，这就使得问题更加复杂。

许多元字符问题都可以采用同一种方法来解决，即使用参数化命令。本节在此基础之上，研究三种其他类型的元字符攻击：路径操作、命令行注入、日志欺骗。

- 使用参数化请求

通过将数据和控制信息分开，可以消除很多元字符攻击漏洞。我们一定要应用好那些本身就拥有这种分隔能力的接口程序，它们允许应用程序提交作为参数值的一系列数据或者一个参数化请求。

- 路径操作

如果允许用户的输入可包含文件系统元字符，例如正斜杠(/)、反斜杠(\)或者小数点(.)，那么攻击者就可以在需要提供相对路径的输入中指定一个绝对路径，或者通过往上移动目录树来将文件系统移动到另一个位置。这种未经授权的文件系统访问称为路径操纵。

- 命令注入

如果允许通过用户输入来指定应用程序所执行的系统命令，攻击者就能在系统中执行他

们想要执行的恶意命令。如果这个输入可包含文件系统或者外壳元字符，攻击者就可以在需要输入相对路径的地方指定绝对路径，或者在应用程序需要执行的命令后面跟随一条恶意命令。未经许可的命令执行权限称为命令注入攻击。

- 日志欺骗

因为日志文件对于系统管理员和开发人员都很重要，所以日志文件也是攻击者的目标。如果攻击者可以控制写入到日志文件中的值，就可以通过在所提供的输入中包括虚构的日志记录来伪造系统时间。如果因为缺少适当的输入验证而使得攻击者可将记录写入到日志文件中，则日志文件的准确性就受到了干扰或者误导，因而降低了日志文件的分析价值。

稍好一点的情况是，攻击者也许只是通过提供包含特殊字符的输入将虚假记录插入到日志文件。如果自动处理日志文件，那么攻击者就可通过改变日志文件的格式来改写这些文件。设计更巧妙的攻击方式还可以歪曲日志文件的统计数据。

伪造的或者被破坏的日志文件可以用于掩盖攻击者的痕迹，或者隐含着另一个恶意行为。如果日志文件包含与安全相关的信息，攻击者可能将日志文件变为常见的错误：如果攻击者在日志文件中写入足够多的误报，使管理员不再对错误进行关注，那么当真正的警报产生时，就不会有人关注。

### 4.4.2　缓冲区溢出

几乎任何一个经常使用计算机的人都知道缓冲区溢出的大名。在软件行业中，通常将缓冲区溢出理解为将大量数据塞入到一个较小缓冲区中所导致的程序漏洞。绝大多数情况下，这是非常精确的理解。当应用程序将数据写入到所分配的内存区以外时，就会发生缓冲区溢出。缓冲区溢出漏洞通常被攻击者用于重写内存中的数据。缓冲区溢出错误非常常见，存在漏洞的应用程序往往会给攻击者提供很大的控制权。因此，缓冲区溢出成为攻击者的常见攻击目标并不奇怪。

#### 1. 缓冲区溢出简介

防止缓冲区溢出攻击的最佳方式是使用强制执行内存安全和类型安全的编程语言。在不安全的编程语言中，例如最广泛使用的 C 和 C++语言，程序员要负责阻止对内存做出不合需要的修改。任何操纵内存的操作都可以导致缓冲区溢出，但实际上，最容易导致缓冲区溢出的错误主要集中在一定范围内的某些操作中。在详细介绍各种缓冲区溢出产生方式以前，先了解一下典型的缓冲区溢出漏洞利用方式。

- 利用缓冲区溢出漏洞

为了解缓冲区溢出漏洞所带来的风险，需要先了解攻击者是如何利用缓冲区溢出漏洞的。在一个典型的堆栈攻击中，攻击者向存在堆栈缓冲区溢出漏洞的应用程序发送包含一段恶意代码的数据。除了在数据中包含恶意代码以外，攻击者还在数据中包含这段恶意代码的初始内存地址。当缓冲区溢出攻击发生时，应用程序将攻击者的数据写入缓冲区中，并且不考虑缓冲区的上限持续写入数据，直到以恶意代码的地址重写函数的返回地址。当这个函数返回时，它直接跳转到存储在返回地址中的值。正常情况下，应该返回到调用函数，但因为返回地址已经被改写了，程序直接跳转到替代缓冲区，继续执行攻击者的恶意代码。为了提高推测出恶意代码正确地址的可能性，攻击者采用的典型攻击方式是在输入的起始处加上一

段 N(非操作性)指令。

对缓冲区溢出攻击最常见的误解就是，只有当缓冲区位于堆栈上时才可能被利用。而事实并非如此，存储在堆上的重要数据同样可以被基于堆的缓冲区溢出攻击改写，而且应用程序控制流也能被攻击者改变。例如，攻击者可改写函数指针的值，当应用程序调用这个函数指针指向的函数时，就会执行恶意代码。最后，即使攻击者不能将恶意代码注入系统中，一种统称为 arc 注入或者 return-into-libc 的利用技术，也可以通过缓冲区溢出来改变应用程序的控制流。

- 缓冲区分配策略

绝大多数防止缓冲区溢出的策略都侧重于如何以及何时检查出某种情况将会导致缓冲区溢出。在讨论这些策略前，先考虑这种检查失败时，将会发生什么事情。其核心就是内存分配问题。如果当前分配的缓冲区空间不能满足操作所需的缓冲区空间，应用程序可执行下列两个操作之一：

1) 保持当前缓冲区大小不变，终止整个程序或者完成部分操作；

2) 动态分配缓冲区，满足操作的缓冲区空间需求。

大型应用程序通常总在某些情况下选择使用静态分配，而在另外一些情况下采用动态分配。程序员需要根据正在进行的任务来选择两种方式之一。无论在特定的程序段中采用何种解决方式，定义一个系统的、明确的内存分配方法，可以使得无论对于人工或者工具来说，检验代码和快速确定其安全性都变得更加容易。对特定类型的操作，一定要保证所选解决方案的一致性。在给定的上下文环境中，要明确已接受的缓冲机制，使得程序员、审查员和工具软件都能明白预期的反应，并能更好确定无意中产生的错误。一致性使得错误更容易被发现。

1) 静态缓冲区分配：在静态缓冲区分配机制中，缓冲区中的内存是一次性分配的，并且这个缓冲区在存储期间一直保持初始分配的大小。这种方法的最大好处就是简单。因为缓冲区在整个生存期内都保持同样的大小，所以程序员易于了解缓冲区的大小，并能确定在其中执行的操作是否安全。给一个缓冲区只分配一次内存简化了缓冲区的操作代码，便于人工和自动代码审查。尽管静态分配方法易于操作，但同时它也有弱点。因为在应用程序编译以前，程序员必须为每个缓冲区选定最大长度，所以静态分配缓冲区的应用程序自然就缺少灵活性。在静态分配机制中，当缓冲区过小而不能满足输入数据的要求时，唯一的选择就是拒绝执行操作，或者截断数据并返回一个错误消息。数据截断操作可能引起大量难以捕捉的逻辑和表达错误，其严重性根据情况发生的环境而定。静态固定大小缓冲区的另一个副作用是，当缓冲区所需的最大字节数远远大于所使用的平均容量时，存在潜在的资源浪费。例如，一个处理电子邮件的应用程序需要分配一个足够大的缓冲区，以便能容纳最大的有效电子邮件地址，虽然在绝大多数情况下这些地址都没有那么长。在大型应用程序中，当许多缓冲区都是部分填充时，这些资源叠加起来也很惊人。

2) 动态缓冲区分配：动态缓冲区分配方法可根据应用程序运行时所需的值，动态分配缓冲区的大小。因为在应用程序编译时不需要确定缓冲区的大小，所以当应用程序所操作的数据在运行时变化很大时，动态解决方案给应用程序提供了更大的灵活性。动态分配还带来了另一个问题。因为动态方法是数据驱动的，意想不到的或者恶意的输入可能导致系统耗尽内存资源。在静态分配方式中，在编译期间就确定了需要接收或者操作多少数据。只有当缓冲

区溢出发生时，这些限制才可能被打破，因此资源耗尽问题并不是程序员考虑的重点。但在动态分配缓冲区时需要限制所接收的数据总量，需要由特定的检查程序来完成这个任务。必须执行明确的、完善的检查程序，以确定在改变缓冲区大小时，没有分配一个不合理的内存空间。

● 跟踪缓冲区大小

C 和 C++内置的帮助程序员跟踪缓冲区大小的功能十分有限，仅返回字符串长度。如果在当前范围的堆栈上为变量分配缓冲区，通过 sizeof 操作能够获取为其分配内存的长度，不过，如果是在堆上为此变量分配空间，则获取的只是指针大小。如果缓冲区含有一个字符串，则要想计算出这个字符串所占内存的大小，可使用计算 Null 终止符前的字节数的方法。唯一的通用解决方案是用单个不同值来跟踪每个缓冲区的当前大小。这样做可以避免通过假定来保证缓冲区操作是安全的。但无论如何，只要某个操作改变了内存分配状况，就需要及时更新所存储的缓冲区长度值。

追踪缓冲区和其大小的最常见方法是，将它们存储在一个复合数据结构中。这个方法基本上就可以保证内存安全。保证内存安全的编程语言追踪每个缓冲区的大小，并且在对缓冲区执行操作以前先比较其大小。如果一种编程语言可以实现这种工作将程序员解放出来，那么无论你的应用程序是否需要以这种编程语言来重新实现，都是值得考虑的。

绝大多数与人工追踪缓冲区大小相关的错误都发生在缓冲区的大小不能够正确维护时。这往往都是因为，在程序中动态更改了缓冲区的大小，却没有更新相关缓冲区大小的值而导致的。一个陈旧的缓冲区大小值比不记录缓冲区大小更危险，因为这个缓冲区上的后续操作会绝对信任已存储的缓冲区大小。最好提供集中的库函数来维护常见数据结构的缓冲区大小。这样做可以仔细审核用于更新这些缓冲大小值的代码，以避免错误。

### 2. 字符串

基本的 C 字符串数据结构(带有 Null 终止符的字符数组)也是错误来源之一，内置的、用于字符串操作的库函数只能使得事情更加糟糕。本节将回顾字符串操作函数的最初设置，然后介绍第二代字符串操作函数(绝大多数这样的函数名中都带有 n 字母)，以及它们所导致的问题。然后探讨使用 Null 终止符来指定字符串长度所带来的问题。随后，讨论字符串导致缓冲区溢出的其他两种方式：多字节字符和格式串。最后研究一些可采用更安全方式来访问字符串的库函数。

● 天生危险的函数

许多 C 字符串函数容易被错误使用。最好是完全避免使用这些函数，而不是小心翼翼地考虑如何才能不出错。尽量不要使用那些天生危险的 gets()、scanf()等字符串操纵函数。

● 有界字符串操作

在 C 和 C++程序中发现的许多早期缓冲区溢出漏洞，都是有字符串操作引起的。当发现像 strcpy()和 strcat()这样的函数存在漏洞时，C 语言的设计者修正了 C 标准，引入这些函数的有边界限制的等价函数，例如 strncpy()和 strncat()。这些有界函数接收一个参数，限制写入到目标缓冲区中的数据总量。

● 有界函数的常见缺陷

相对于无界函数而言，有界字符串函数更安全一些，但仍可能产生错误。下面就是程序

员在使用这些有界字符串时，经常会遇到的错误：

1) 因为边界是根据源数据的大小(而不是目标缓冲区的大小)来确定的，所以可能导致目标缓冲区溢出。

2) 目标缓冲区中没有添加一个 Null 终止符，这通常是因为 off-by-one 错误而导致的。

3) 因为边界指定为缓冲区的总容量而不是其剩余空间，所以导致目标缓冲区溢出。

4) 因为目标缓冲区没有设置一个 Null 终止符，所以，函数在目标缓冲区的第一个零字符所在位置后写入数据，从而导致应用程序在内存的任意位置写入数据。

为避免发生这些问题，首先提出的建议就是，避免使用两个最常被误用的有界字符串操作函数：strncpy()和 strcnat()。然后探讨关于字符串截断导致发生错误的更加广泛的主题，因为即使正确调用有界函数，这些错误也可能发生。

- 保留 Null 终止符

在 C 语言中，字符串正确与否依赖于是否正确地使用 Null 终止等；没有 Null 终止符，就不能够确定字符串的大小。这种依存关系式相当脆弱的，因为，它依赖于字符串的内容来保证与字符串相关的操作能否正确运行。

字符串终止错误可以轻易导致缓冲区溢出和逻辑错误。这些问题常常比较隐蔽，因为它们的发生似乎总是非常确定的，往往依赖于程序执行时的内存状态。在一个存在很多漏洞的应用程序执行期间，位于一个没有正确终止的字符串变量之后的内存可能恰好是空的，所以完全掩盖了错误，但在同一个应用程序的不同执行中，位于这个字符串后的内存也许是非空的，导致与这个字符串相关的操作完全错误地运行。在复杂程序中，依赖于内存运行状态的漏洞难以查找，往往直到程序在实际运行环境中运行时才会发现，因为真实环境与测试环境相比而言，程序运行的时间更长，所处的环境更复杂。

- 字符集、表达式和编码

字符集是打印字符的集合，通常对应于在书写语言中使用的字符。随着各种软件程序遍及世界各地，已经定义出各种不同的字符集来满足使用不同语言的人们的需求。Unicode 标准定义了一个通用字符集，包含了全世界每一种主要的书写符号以及一套字符编码格式来存储、操作和共享各应用系统之间的文本信息。

提交给计算机的字符采用了字符编码格式，这些格式制定了整数值和打印字符之间的一个映射。编码方式基本上分为两类：固定宽度和可变宽度。固定宽度编码使用固定的位数来代表每个码点，固定宽度编码具有一致性，更便于计算机和程序员操作。但是如果在某些字符串中只使用了一少部分码点，那么固定宽度编码方式就不够高效。采用可变宽度编码，某些字符用较少的位来表示，而某些字符用较多的位来表示就可以克服这个问题。

- 格式串

当允许用户输入影响某些字符串格式化函数的格式串参数时，可能会发生格式错误。本小节简单讨论格式串漏洞的成因，概述产生这些错误的几种方式。

从历史上看，新的安全漏洞并非总是频繁产生。与病毒研究者不同，软件安全研究这并不需要每天或者甚至每年都发现新的漏洞，但一旦发现了可能需要几年的时间来克服。关于格式串漏洞来说也是这样的情况，早在 1999 年就被广泛认定了。软件行业已经用了十几年的时间来讨论缓冲区溢出漏洞，但当 2000 年格式串漏洞被广为利用时，C 和 C++中经常出现的一个常见弱点突然引起了公众的注意，大量软件受到影响。

C 和 C++中的字符串格式化函数，例如 printf()，都设计得尽可能灵活。因为有效的格式串并不要求必须包含指令或转换说明，所以，格式串参数可用于处理不需要格式化的函数。格式串以外的任何参数、对应格式指令或者转换说明符都是可选的，这种灵活性本质上是一个类型问题，会让程序员在编写时采用一些简便写法，虽然这样操作表面上不会带来任何错误，因为标准字符只是简单地通过格式串传递，没有任何改变，但它是导致格式串漏洞的最常见方式之一。

虽然漏洞利用方式有很多而且很复杂，但如果在下列原则中，选择一个可在应用程序环境中执行最严格的限制，那么绝大部分格式串漏洞是可以避免的：

1) 对任何接受格式串参数的函数，都只传递静态格式串，

2) 如果只传递某个固定的静态格式串的限制太严格，可定义一个有效的格式串集合，然后从这个安全的集合中选择格式串。

3) 如果应用程序必须要求格式串包括从应用程序外部读入的输入，那么对于从应用程序外部读入的、将要包含在格式串中的值，应执行严格的、基于白名单的输入验证。

● 更完善的字符串类和类库

就像 C 语言本身一样，C 语言中的字符串更多考虑了有效性和简单性，而不是稳健性和安全性。使用 Null 终止字符串具有更高的内存使用效率，却容易出错。C 字符串的替代表示方法不需要将防止缓冲区溢出作出一个功能来强调其安全性。如果使用 C++语言，则使用标准 STL 命名空间中所定义的字符串表示法，即 std::string。std::string 类在底层的字符串表示外添加了一个抽象层，提供了执行绝大部分字符串操作的方法，并且不用冒着引入缓冲区溢出漏洞的风险。

虽然存在大量用于替换 C 语言类库的其他字符串处理类库和表示方法，但是并没有哪一个得到广泛使用。绝大部分程序员仍继续使用 C 语言本身带的字符串，并且寄希望于它们可以避免错误。选择一种能够更容易地避免缓冲区溢出的字符串表示方法，就可以更容易地编写出安全的应用程序，并且能带来其他好处。C 语言中 Null 终止字符串在设计时侧重于编程方便，转为寻求其他不同的字符串表示方法时，可以注意到程序运行性能的提高，具体指标根据所执行的字符串操作类型而异。

### 4.4.3 缓冲区溢出伴随的问题

缓冲区溢出的原因不全是字符串操作错误，即使选用更好的字符串操作函数也无法保证不再发生缓冲区溢出。当一个整数值超出分配空间所能表示的范围时，就会发生整数溢出错误，而它通常就是攻击者进行缓冲区溢出攻击的前奏。最常见的缓冲区溢出是数字类溢出，这是因为程序中操作最多的是数字类数据。同样，不是每一种应对缓冲区溢出攻击的对策，都是以明智的字符串操作形式来实现的。

本节首先介绍整数溢出错误以及其导致缓冲区溢出攻击的一般原理。然后讨论程序运行期间用于减少缓冲区溢出漏洞的方法，包含其优点和缺点。本节分为两个部分：

1) 整数：如果一个操作所传递的变量可能超过最大或最小值范围，就可能是缓冲区溢出的前兆。

2) 运行时保护：采用类型与类型均安全的语言进行编程是保护防止缓冲区溢出的最好方法，如 Java 或 C#，或者选择更安全的 C 语言。另一种可选的方案是，在编写完应用程序之

后，插入运行时缓冲区溢出保护程序，但该方法解决不了所有问题。

### 1. 整数

因为所有内置整数类型都用一个固定位数表示，所以其表示范围有限。有时，程序员忽略了这个事实，认为整型变量和数学中的整数一样，没有上下限。实际上，与数学中的整数不同，程序中的变量范围固定，当值超出最大或最小值范围时，就会“回绕”；比如，一个非常大的正数变成一个非常大的负数，反之亦然。

如果攻击者利用这种回绕错误，应用程序就会面临一个整数溢出攻击。整数溢出能导致许多问题，但在 C 和 C++中，最常见的是将整数溢出作为缓冲区溢出利用的“先头部队”。当使用可能发生回绕的变量来分配内存、限制字符串操作范围或作为指向缓冲区的指针时，可能发生缓冲区溢出。在 Java 中，也可能发生整数溢出，但因为 Java 强制执行内存安全性策略，所以整数溢出漏洞相对不容易被利用。

本节主要讨论整数溢出和数字转换错误发生的各种形式，建议在通常情况下采用什么策略来避免这些问题。然后研究一些关系密切的类型转换错误。最后，针对最容易遭到攻击的几种情况，示范了采用哪些技术来预防和检测这些问题。

- 回绕错误

回绕错误是整数溢出的典型示例，在一个整数值超出其数据类型所限定范围的情况下，就会产生回绕错误。就像回绕错误会引起整数上溢，认识到这点很重要。

- 截断和符号位扩展

当数据一个整数类型向一个比自己位数少的数据类型转换时，就会产生整数截断错误。C 和 C++通过截去高阶位来进行这种转换。当一个有符号整数从较少的位数转换为较多的位数时，多余位数将被填充以保持原有符号，与前面截断操作的方向正好相反。这就意味着，如果一个负数被转换成一个更大的数据类型，它的有符号值将保持不变。但它的无符号值将急剧增加，这是因为它最重要的数据位会被设置值。

- 有符号数和无符号数之间的转换
- 检测和防止整数溢出的方法

没有简单的、容易遵循的建议能完全消除整数溢出和类型转换错误，像许多缓冲区溢出漏洞一样，整数溢出几乎不可能被完全防止。据此，提供一些通用的规则，用于限制整数溢出所引起的风险。

### 2. 运行时保护

避免缓冲区溢出攻击的最好方法是，采用一种根本不允许这些错误发生的编程语言，比如 Java 和 C#。如果已经编好的程序很庞大，那么改变语言通常是不可取的。

在程序编写完毕之后安装补丁程序似乎更容易实现。但是，这样做的副作用就是，这些补丁程序并不能实现同等级别的保护。换句话说，它们不能真正解决问题。

- 更安全的编程语言

尽管频繁公开出现各种漏洞似乎说明编程语言还不够安全，但是已经存在非常完善的缓冲区溢出方案。Java、C#、Python、Ruby 和其他一些编程语言，确实消除了缓冲区溢出错误的可能性。但这些语言提供的缓冲区溢出保护措施，是以相当高的代价换来的。本节概述了

安全性和其他需要的编程语言特性之间不可避免的折中，比如系统性能和灵活性。对于每个具体的应用程序，并不存在唯一一个正确答案；只能尽力提供合理解决问题所需的必要信息。

- 更安全的 C 语言
- 动态缓冲区溢出保护

寻找解决难题的简单办法是人类的本性；在预防缓冲区溢出攻击方面也不例外。存在很多没有解决根本问题的检测或预防缓冲区溢出的示例，所有的这些方法都失败了。不要被权宜之计带来的满足感所迷惑。本小节讨论下列动态保护类型，侧重于改变程序二进制表示法或运行环境：不可执行的内存段；编译时工具；虚拟执行环境；更健壮的系统类库。

### 4.4.4　错误和异常

安全问题的发生常常是因为攻击者找到了某种违背程序员初衷的途径。程序员通常不会耗费精力完全考虑错误条件和异常情况，但是这种忽视的地方将使攻击者具有利用程序自身失误发起攻击的能力。本节将主要考虑常见错误和异常处理方法所隐含的安全问题。本节所讨论的大部分错误不会像缓冲区溢出或 SQL 注入一样直接导致可利用漏洞。但它们会为随后发生的安全问题提供必要条件。

通常，编写应用程序所用的计算机语言确定了这个应用程序用于发现及处理异常情况的方法。C 语言提供错误代码作为函数的返回值。Java 使用已检测异常。C++混合使用返回值和未检测异常。不管采用哪种方法，不完善的处理机制通常会导致资源泄漏。错误处理代码常常也可能导致日志或调试软件问题。

#### 1. 利用返回代码处理错误

利用函数返回值表示函数是否执行成功，是一个简单明了的方法。但是，这个方法伴随着大量副作用：

1) 易于忽略错误(仅忽略函数返回值)；

2) 错误处理代码和错误信息交织在一起，从而程序变得难以理解。如果错误处理逻辑中又夹杂着处理正常情况的逻辑，就更容易忽略错误了；

3) 因为没有任何约定俗成的通用方式来表示错误信息，所以程序员必须研究所调用的每个函数的错误处理机制。

- 检查 C 语言中的返回值
- 检查 Java 中的返回值

#### 2. 管理异常

异常可以解决很多错误处理问题。虽然，只要简单地省略查找函数返回值的代码，就可以容易地忽略函数的返回值，但忽略一个已检测的异常则需要做相反的工作——程序员必须为此编制专门的代码。异常分为允许遵循正常路径的代码和处理异常情况的代码。

异常包括已检测的异常和未检测的异常。其不同之处在于，编译器是否将使用静态分析以确保异常被处理。假如一个方法声明抛出一个已检测的异常，所有调用它的方法就必须处理这个异常或者声明同样也抛出这个异常。这就迫使程序员考虑所有可能会发生已检测异常的地方。Java 编译器强制执行这些与已检测异常相关的规则，Java 类库提供了自由利用已检

测异常的功能。Java 类 java.lang.Exception 就是一个已检测异常。

未检测异常可以不必声明和处理。C++中的所有异常都是未检测的，这就意味着程序员可以完全忽略可能会出现异常这个事实，其编译器也不会发出警告。Java 也提供了未检测异常。

带有未检测异常的风险是：程序员可能意识不到在给定上下文环境中可能会发生异常，可能会忽略适当地处理错误。

- 捕获顶层的所有东西
- 消失的异常
- 只捕获需要处理的异常
- 控制已检测的异常

### 3. 防止资源泄漏

未能释放资源(包括数据库对象、文件句柄和套接字)可能会导致严重的性能问题。要想找出这些问题的原因比较困难。这种情况通常是在不寻常环境下或系统负荷太重时偶然发生的，所以很难根据生成的错误信息查找到资源泄漏的位置。一个简单的示例：close()调用完全被忽略。但更常见的是，这个问题不得不处理极少使用的代码路径，它们通常包含错误的条件或异常。

在 C、C++以及任何其他依靠人工进行内存管理的语言中，对分配内存必须像释放其他资源一样被释放。不管资源中包含什么东西，释放的方法总是相同的。下面给出了与对分配内存、文件句柄、数据库连接相关的示例，但是同样的模式可应用于其他所有资源。

不恰当地管理资源所导致的安全性后果都是一样的。假如程序不能正确管理资源，那么攻击者就很容易对程序发起拒绝服务攻击。但引入这个主题，主要是因为在执行安全代码审查时，经常遭遇资源管理问题。由资源泄漏引起的问题类似于安全性问题，因为运用传统的测试方法无法识别和跟踪它们。

不论你把资源泄漏分为安全风险问题(因为资源泄漏可能会导致拒绝服务攻击)，还是一种普通的质量问题(因为其可能与性能相关)，其解决方案是相同的：建立资源管理体系。因为错误处理代码与资源泄漏之间的关系，错误处理模式必须在各种条件下(不仅是某些期望的情况)清晰指明资源管理方式。

### 4. 日志记录和调试

日志记录和调试都能深入了解在程序执行过程中、特别是在发生错误或出现预料外的情况时，发生了什么。本节将阐述创建一个持续日志记录操作和将调试辅助程序从二进制代码中分离的优点。

利用一个集中日志结构。集中框架结构有利于以下工作：

1) 提供一种通过日志记录反映出来的一致和统一的系统视角；

2) 方便修改，例如：将日志移到另一台机器上、将记录到文件的日志改为记录到数据库，更新有效性或私有标准。

不论选什么，必须确保在代码中一直运用一样的日志记录机制。尽量不要用 Sytem.out 和 System.err 进行特别记录，因为它们很难确认正确操作还是日志记录约定更新。接下来将讨论良好日志记录的一些基本要求：带时间戳的记录条目、记录所有重要行为、可控地访问

记录数据。

谨慎地将调试代码和系统分离开来，以便它绝对不出现在产品部署中。每个人都增加代码容易泄露系统的额外信息。Knuth 指出：最有效的调试技术是在程序自身中设计和构建的调试技术，当今在许多最好程序员的程序中接近一半的内容都用于简化另一半程序的调试过程。前一半内容……将最终被抛弃，即使这样，其性能也可能更好。问题是，如果这种扩展行为离开调试环境并进入应用环境中，将对系统的安全造成潜在危害。调试代码没有像程序其他部分一样接收同等级别的检查和测试，并且编写时也没有从稳定性、性能和安全方面考虑。

“后门”访问代码是调试代码的一个特别情况。“后门”访问代码允许开发者和测试者通过终端用户不能使用的方法访问应用程序。在孤立环境中或应用程序被配置到应用环境前，“后门”访问代码是应用程序必要的测试组成部分。但是，假如“后门”访问代码被发送到实际产品中，攻击者就可以使用程序设计和调试过程中没考虑到的方法攻击程序，经常可以绕过运行过程中的安全机制。当这种类型的调试代码意外存在于产品中时，这个应用程序无意之中开放了交互模式。因为这些“后门”进入点在设计或测试时没有被考虑进来，且在应用程序期望的操作条件之外，它会引起安全风险。

遗忘调试代码最常见的示例是，Web 应用程序中的 main()方法。尽管这是在产品开发过程中可接受的惯例，但 J2EE 应用程序的部分类不应该定义 main()方法。其他“后门”访问机制的示例通常包括，在应用被配置前允许综合测试或简化管理过程的 Web 访问节点。如果它们被忘记了，则存在更大的风险，因为它们可以被 Web 上的任何人访问，而不仅是可以局部调用 main()方法的用户。

确保那些没用的、临时的、备份的文件不出现在产品中。备份文件是软件开发过程中的附属品。创建它们的原因有很多，例如，程序员在抛弃先前工作前想验证其他新功能，或者测试者需要用给定组件的不同版本测试不同场景或不同模块。不论创建临时文件的原因是什么，攻击者可以通过临时文件在攻击产品前对它的大致构型有一个粗略了解。因为临时文件反映了之前代码或设置，所以它们也最容易遭受安全攻击或出现其他错误。应用程序框架也会用不同的方式对待临时文件，例如，名为 index.jsp 的文件将被编译成 Java 服务器端页面，而名为 index.jsp.bak 的文件可能在源代码中以文本的形式出现。

自动 Web 攻击工具通过识别暴露在整个站点中的文件名的方式来搜索备份文件。例如，假如一个站点包含文件 welcom.jsp，攻击工具将请求 welcome.jsp.bak、welcome.jsp.txt、welcome.jsp 和 welcome.jsp.zip 等文件。确保查明所有这种类型的最简单方法是，确定不配置备份文件。从输入确认技术中获得线索，建立一个所发布文件的白名单。当然，也非常有必要对应用程序的发布版本进行测试。不要把工作都留给测试工具来猜测哪种垃圾文件已经丢掉，确保所有的配置文件都有合适的文件扩展名是部署过程的一部分。

## 4.5　本章小结

本章主要介绍了代码安全静态分析的基本概念以及代码审查中的静态分析，其中，代码安全静态分析基本概念部分还对静态分析存在的局限性和其所能解决的问题进行了重点阐

述。另外，本章还着重介绍了静态分析的过程和该过程中经常出现的问题，静态分析过程包括建模、分析算法、规则以及报告结果，而该过程中经常出现的问题则包括处理输入、缓冲区溢出及其伴随的问题、错误和异常等。静态分析在软件测试中占有十分重要的地位，它不仅关系着程序员编程的规范性，更关系着整个程序的安全性。

# 第5章　软件安全动态渗透测试

## 本章导读

渗透测试是受信任的第三方进行的一种评估网络安全的活动，它是一个通过各种手段攻击企业网络以找出系统中存在的漏洞，进而给出企业网络中存在的风险的过程。通过模拟现实的网络攻击，渗透测试证实恶意攻击者有可能获取或破坏企业的数据资产。本章将介绍渗透测试的相关知识。

应掌握的知识要点：

- 渗透测试的概念；
- 建立测试计划；
- 主机侦查；
- 攻击 Web 服务器；
- 数据库攻击；
- 口令破解；
- 会话劫持；
- 木马和后门的运用；
- 常见服务器渗透；
- 缓冲区溢出；
- 拒绝服务攻击；
- 软件漏洞。

## 5.1　软件安全动态渗透测试的概念

在当今的信息时代，由于 Internet 连接是公开的，因此极大地增加了保护网络和计算机系统安全的难度。各种网络攻击手段层出不穷，企业和机构为降低各种攻击所带来的风险，更注重对自己的系统进行专业的渗透测试，找出其中存在的安全漏洞和薄弱环节。那么，什么是渗透测试呢？简单地说，渗透测试是由受信任的第三方进行的一种评估网络安全的活动，它是一个通过各种手段攻击企业网络以找出存在于系统中的漏洞，进而给出存在于企业网络中的风险的过程。通过模拟现实的网络攻击，渗透测试证实了恶意攻击者有可能获取或破坏企业的数据资产。

### 5.1.1　渗透测试概述

黑客这个术语的现代意义，起源于1960 年的麻省理工学院技术模型铁路俱乐部。这个俱

乐部设计比例较大、细节逼真的火车模型，而“黑客”被用来称呼那些发现了聪明技巧的俱乐部成员。从那以后，“黑客”这个术语用来描述从计算机迷到天赋程序员之类的人，他们的共同之处是对计算机系统和网络都有自发的探索之心。另外，开源软件开发者也经常认为自己是黑客，并且把此称谓当作一种尊称。渗透测试人员是一个有道德的黑客，他被雇佣来寻找公司网络的漏洞，以便评估数据安全特性。攻入网络的道德黑客小组被称为老虎队。老虎队是一组程序员或用户，他们自愿或被雇佣来发现新开发的软件或网络系统中的错误和安全漏洞，或者弄清原有计算机网络的安全被瓦解的原因。在美国军队中，老虎队是一支被分配了突破敌方基地或限制区安全线任务的特殊部队。

渗透测试人员能够进行三种类型的测试：

1) 黑盒测试：渗透测试人员不具备公司网络的任何先验知识。例如，进行外部的黑盒测试时，测试人员拿到手的信息仅为网站的地址或 IP 地址，然后被要求像恶意黑客那样破解指定的网站。

2) 白盒测试：渗透人员已经具备内部网络完整的知识。他们在测试之前已经得到了一张网络拓扑图或一张操作系统和应用程序列表，并可从中获取企业网络的完整知识。尽管这种方式不是外部攻击的最佳表示，但它却是最精确的，原因在于它表达了最糟的情况。

3) 灰盒测试：测试人员模拟内部雇员。他们得到了一个内部网络的账户，并拥有了访问网络的标准方法。这项测试用于评估来自企业内部雇员的攻击。

渗透测试人员要测试漏洞和威胁。漏洞是能够用来破坏安全策略的弱点、设计或实现错误；威胁是一种能引起破坏的潜在安全侵害，如暴露敏感数据、篡改数据、破坏数据及拒绝服务等；而安全则关注避免威胁的资源保护。威胁可能破坏三个方面的特性：机密性、完整性和可用性，简称为 CIA(Confidentiality，Integrity，Availability)。其中，机密性威胁是指数据被非授权查阅的风险，完整性威胁是指数据被非授权用户篡改的风险，可用性威胁是指用户不能使用服务或网络资源的风险。

### 5.1.2　渗透测试阶段

渗透测试可划分为下述 5 个阶段：侦察、扫描、获取访问、维持访问和擦除证据。

在侦察阶段，测试人员尽可能多地收集所选目标的信息。侦察分为主动侦察和被动侦察两种形式。在主动侦察攻击中，测试人员使用诸如 nslookup、dig、SamSpade 的工具探测目标网络，以获取诸如 IP 地址范围这样的内容。在被动侦察攻击中，测试者使用诸如新闻组或招聘帖子这样的公开可用信息来发现有关公司技术的信息。

在扫描阶段，测试者使用诸如 Nmap 这样的工具扫描开放端口，以获取网络的特征。这个阶段的目标是确定运行在目标主机上的服务。也正是在这个阶段，测试人员完成操作系统的指纹识别，即通过匹配目标主机的操作系统特征，确定该操作系统的类型。

扫描阶段的工作也包括漏洞扫描。漏洞测试的目标是试图利用被发现的漏洞，而当成功利用后，就获取了对目标主机的访问。通过安装后门木马程序，测试者能反复进入目标系统，这样就实现了维持访问的目的。

测试的最后一个阶段是擦除证据。有道德的黑客希望了解他们能否擦除可能记录了他们在目标网络中活动的日志文件。由于很多攻击就是在悄无声息进行，因此，评估日志中能够记录什么攻击以及擦除这些日志的难度也是一件重要的事情。另外，在擦除日志之前一定要

确保得到了这种授权。如果测试者不能证明他们进行了日志擦除，那么擦除这样的日志文件就会导致责任问题。如果没有授权擦除日志，那么可以测试事件通知过程，并向客户了解是否得到了适宜的通知。

# 5.2　建立测试计划

常言道，不打无准备之仗。与所有大型项目一样，渗透测试的成功也来自严谨的系统规划。渗透测试并不是通过随机运行几个工具就完成了安全评估。渗透测试需要建立一个系统的、可以一步步精确执行的计划，这个计划必须精确地描述要做什么、什么时候做以及如何去做。

## 5.2.1　分步骤的测试计划

完整的渗透测试都应该包括下述步骤：

1) 侦察——收集目标网络信息的最初阶段；
2) 列举——查询活动系统，从而抓取网络共享、用户及特定应用程序信息的过程；
3) 获取访问——实际渗透过程；
4) 维持访问——测试者将后门程序放入到被利用系统中，以便未来使用；
5) 抹去踪迹——删除日志文件项、擦除进入系统痕迹的过程。

在开始第一步工作之前，还要和客户或管理层(如果进行内部测试的话)完成下述任务：缩小项目范围；确定是否使用社会工程；决定是否允许使用会话劫持；就木马和后门软件的使用达成一致。

## 5.2.2　开源安全测试方法指南

安全与开源方法协会的各位专家通力协作，制定了“开源安全测试方法指南”(OSSTMM, Open-Source Security Testing Methodology Manual)，为渗透测试者设计安全审计方法提供了基本的参照。

OSSTMM 针对诸如信息安全、过程安全、互联网技术安全、通信安全、无线安全、物理安全等安全评估问题进行了重点阐述。虽然其包含范围比渗透测试要广，但它依然可以作为渗透测试过程中的初始框架。数据收集完成后，就可以进入评估阶段了。

## 5.2.3　文档

如果不能向客户或主管交付有形的东西，那么渗透测试将变得毫无意义。结果报告应该详细列举测试的成果，另外，当所检测的系统被判定为高风险系统时，测试人员应当为客户编写专业的建议。

一般情况下，渗透测试报告由下述章节构成：摘要、项目范围、结果分析、小结以及附录。

# 5.3　主机侦察

恶意黑客想要进行有效的攻击，首先要做的就是进行有价值的侦察。因为如果目标运行着 Microsoft 服务器，那么对它发起与 UNIX 漏洞利用相关的攻击是毫无意义的。在前期侦察上花些许时间将节省渗透攻击期间的大量时间。有时，恶意攻击者在突破目标的安全防线之前可能要花数月时间来进行仔细侦察。尽管渗透测试人员没有像恶意攻击者那样有很多的时间，他们依然知道侦察的价值。主机侦察的目标是要发现下述信息：目标网络上主机的 IP 地址，目标系统上可访问的 UDP 端口和 TCP 端口，目标系统上使用的操作系统。

本节将介绍如何使用各种信息收集技巧发现目标网络上的活动主机。利用端口扫描工具，我们也将学习到如何确定目标主机上运行的操作系统及开放的 TCP 和 UDP 端口。最后将介绍监测和预防侦察技术的最佳实践。

## 5.3.1　被动主机侦察

用来发现目标网络上主机信息的侦察方法有两种：被动侦察和主动侦察。本节主要介绍被动主机侦察。

被动侦察是从开放的信息源中收集数据。开放信息意味着这些信息可自由地从外部访问，且查看开放信息源的信息是完全合法的行为。对于这些信息的保护，企业几乎做不了什么，但如果企业愿意，依旧可以找到一些可用的、保护企业公开信息的方法。

常见的开放信息源包括：公司网站，电子数据收集、分析和检索系统档案(适用于公众上市公司)，NNTP USEET 新闻组，用户组会议，商业伙伴，垃圾侦察以及社会工程。

## 5.3.2　主动主机侦察

尽管被动侦察方法很有效，但它经常要花费大量的时间，而且并不一定会生成最精确的结果。在主动侦察中，使用技术工具发现目标网络上活动主机的信息。然而，主动侦察也存在缺点，主要是这种侦察方式容易被发觉。以小偷盗窃为例，一个是他在准备动手的房子周围转悠(被动侦察)，另一个是趴在房子的每一个窗户上窥探房间里有什么东西(主动侦察)。显然，窥探窗户的小偷比走过房子旁边的小偷更值得怀疑。总之，主动侦察揭露了更多的信息，但也更容易被发现。

主动主机侦察中常用的工具包括：NSLookup/Whois/Dig lookups，SamSpade，Visual Route/Cheops，Pinger/WS_Ping_Pro 等。

## 5.3.3　端口扫描

知道目标网络上有哪些主机能够公开访问后，就需要进一步确定这些主机上开放了哪些端口，而要完成这个任务就需要使用端口扫描。端口扫描是一个通过扫描主机确定哪一些 TCP 和 UDP 端口可以访问的过程。

现有的绝大多数应用程序都运行在 TCP 或 UDP 协议之上。这些协议是众多应用程序使用的传输机制，如文件传输协议(FTP)、简单邮件传输协议(SMTP)、动态主机配置协议(DHCP)及超文本传输协议(HTTP)等。TCP 是一种面向连接的协议，也就是说，它通过在主机之间建

立连接来提供可靠的访问。与此相反，UDP 是一种无连接协议，它不提供可靠性保证。

TCP 类似于寄挂号信，收件人必须签字表示收到了信件，信件的传递才算成功，从而保证了可靠性。与此相反，UDP 类似于普通信件，它并不保证信件会被正确地送达。UDP 应用程序(比如 DHCP)依靠应用程序本身在必要时提供可靠性。而使用 TCP 的应用程序(比如 FTP)运用了内置在 TCP 协议中的可靠性来提供可靠的数据传输。

端口扫描有多种类型，主要包括：TCP Connect()扫描，SYN，NULL，FIN。

### 5.3.4　扫描检测

由于主机和网络的扫描通常是其他漏洞利用和攻击的前奏，因此它们不可能悄无声息地进行。Nmap 是渗透测试人员进行端口扫描的一个有用工具，它具有使用灵活、功能强大的特点，而构造一个抵御 Nmap 扫描的堡垒需要几个安全部件。本节将主要讲述这些安全部件，为进一步学习利用入侵检测系统监视和检测通过 Nmap 工具扫描网络和主机的方法打下基础。

#### 1. 入侵检测

入侵检测系统(IDS，Intrusion Detection System)与家庭安防系统(防盗警报器)类似，防盗警报器监视进入或侵入家庭和办公室的不速之客，IDS 则记录进入网络的警报。但是，与家庭安防系统不同的是，我们可以配置 IDS，使其能够真正抵抗 TCP RST 和 SHUN 命令，终止恶意攻击对网络的进一步侵入或破坏。IDS 通常部署在能检查到尽可能多的流量的地方。

#### 2. 异常检测系统

异常检测系统(ADS，Anomaly Detection System)也称为基于轮廓的检测系统(Profile-Based Detection System)，它设计用于观察用户或网络的概貌。例如，如果异常检测系统观察到网络流量利用率从正常时期的 30%突然提高到 90%并且持续一段时间，那么它就会报警。

#### 3. 滥用检测系统

滥用检测系统(MDS, Misuse Detection System)使用模式匹配方法。这样的系统包含了用于匹配网络流量的数百种模式和特征标志。滥用检测系统与标准的磁盘反病毒软件的思想方式十分相似：反病毒软件扫描磁盘驱动器，寻找磁盘或文件中与病毒特征相似的模式；滥用检测系统不是读取磁盘文件，而是直接读取电缆上的数据帧和数据包。当前的大多数检测系统都采用了这种工作方式，但它们的特征库随着新的攻击方式的出现很快就会过期，因为新攻击的特征还没有收录到此类系统的特征数据库中。

#### 4. 基于主机的入侵检测系统

基于主机的入侵检测系统安装在本地主机上，用于检查本地系统上的一些重要安全因素，包括系统调用、审计日志、出错消息及网络流量。基于主机的入侵检测系统的优点是它们能为特定系统提供保护和报警。然而，这类系统并不是设计用于对整个网络提供保护，而只是对主机提供保护。

### 5. 基于网络的入侵检测系统

基于网络的入侵检测系统专用于完成一件任务——监视整个网络，这样的系统包括 Cisco 4200 系列的设备。它们被放在特定检查点并使用专用端口来监视传递到网络中任何主机的网络流量。

### 6. 网络交换机

在网络集线器出现不久，交换机就进入了市场。交换机提供了与集线器相同的星形拓扑结构，但它不是把所有计算机就连接在一条总线上。当计算机之间进行通信时，交换机监视第 2 层的帧，并形成一个 MAC 地址表。它通过创建计算机到指定端口的内部映射表，增加交换机的性能。当计算机要跨越交换机进行通信时，交换机把这些帧转发给包含目标主机的特定接口。

流量仅被转发到它所需要的地方。基于这种基本设计，低价格的交换机难以有效地安装 IDS。昂贵的可编程交换机通常支持被称为交换端口分析器(SPAN，Switched Port Analyzer)的端口或端口监听。SPAN 功能让网络管理员能够选择一些特定端口，并将它们的所有流量都复制出来用于监测。反过来，这些端口也是 IDS 要连接的地方。

# 5.4 Web 服务器攻击

互联网已经深入到了生活中的各个角落，人们可以从网上购买到自己所需要的几乎一切东西，如足球、股票、汽车、电子产品、图书、门票、保险等。

今天，人们足不出户，就能够完成下述任务：

1) 购买能够想象得到的几乎所有商品和服务，如汽车、电脑、图书、家政服务等；

2) 完成复杂的金融交易，如银行服务、股票买卖、基金买卖等；

3) 寻找人类已知的每一个主题深入透彻的信息；

4) 搜索海量信息，如果在 Google 中输入一个比较常见的词汇，就会找到成千上万出现该词汇的网页；

5) 体验几乎无限的数据媒体内容，如电影、音乐、图像、电视等；

6) 得到各种各样的软件，并且大部分是免费得到，如从操作系统到文字处理程序等；

7) 以实时方式与任何人进行通信，包括使用基于 Web 的电子邮件、语音电话及聊天工具等。

伴随众多服务和便利网上购物的是日益增加的安全问题。尽管网上购物的安全风险并不比到商店中购物大，但联机攻击也在日益增加。现在恶意黑客能够在自己的家中发起攻击，而且人们很难发觉。网上攻击通常难以被监测到，即使被监测到，也很难追踪到攻击的发源地。出于这些理由，企业雇佣渗透测试人员来评估其网上资源的安全性。这种测试通常包括攻入网站测试以及评估攻击过程被监测到的即时性。

## 5.4.1 Web 语言简介

互联网的发明引发了一次技术爆炸，导致了一场提供优势 Web 服务器和后台语言的竞赛。

Web 技术中的主要语言当属 HTML，由于已比较成熟，所以发展潜力不是很大。Web 技术竞争最激烈的是服务器技术和脚本语言这两个领域。微软、Sun 等老牌软件企业纷纷推出了自己专属的 Web 技术，而且这些技术最重要的特征是跨平台，这就给保证网站的安全带来了不小挑战。另外，渗透测试人员或 Web 黑客通常能在 Web 网站上找到演示代码，甚至找到设计欠佳、不安全的网站。由于缺乏对 Web 语言、设计及服务器配置的完整知识，网站每天都面临着被攻击和破坏的可能。

1. HTML

HTML(Hypertext Markup Language，超文本标记语言)是当今格式化 Web 页面的事实标准。当我们打开一个 Web 页面时，就会看到不同颜色、大小的文字、按钮、列表框、图片及到其他页面的链接等。所有标准的 Web 页面都按照 HTML 预定义结构来编写格式。当我们使用如记事本这样的简单编辑器打开 HTML 文件时，就可以看到用于格式化 Web 页面的源代码。

HTML 是用于定义 Web 页面颜色以及其他特性的一种语法，它最初于 1989 年由 Tim Berners-Lee 发明。HTML 基于更古老的语言 SGML 进行定义，并简化了其中的语言元素。这些元素用于告诉浏览器如何在用户的屏幕上显示数据。HTML 没有做什么特别复杂的东西，也没有提供闪烁移动内容的标记，所有内容都是静态内容。

2. DHTML

DHTML 是 Dynamic HTML 的缩写，意思为动态 HTML。它支持控制客户端浏览器中的 Web 页面，以此对标准的 HTML 进行了扩展。例如，当访问某个网站时，如果看到这个网站上的网页能够更换图像、弹出对话框、当鼠标指针移到上面时链接改变颜色等，那么基本上可以肯定这个网站使用了 DHTML。由于 DHTML 能够增强用户的体验，因此几乎所有大中型网站都使用了它。

3. XML

与 HTML 一样，XML(Extensible Markup Language，可扩展标记语言)也是从 SGML 标准中导出的。XML 用于描述、存储和交换数据，这样，就使得各种类型平台上的数据具备了相互理解的能力。在 XML 出现之前，当两个系统交换数据时，系统或应用程序以只有这两个系统能够理解的格式发送数据。这类数据格式的一个典型示例是以逗号分隔值的文件格式(Comma-Separated Value，CSV)。CSV 格式的文件采用逗号或 Tab 字符分隔原始数据。当我们打开一个别人创建的 CSV 文件时，如果没有人为我们说明这个文件，那么我们就很难理解这个文件中数据的意义。类似 CSV 格式的数据生成起来很容易，但不具备可扩展能力。而在 XML 中，由于其对数据进行了描述，因此该格式的文件易于被理解。XML 格式包含两部分内容：一部分是包含数据的文档，另一部分是描述文档中所存储数据的数据类型的文档类型定义(DTD，Document Type Definition)。

4. XHTML

XHTML 是一种为适应 XML 而重新改造的 HTML，是 HTML 4.0 的后续版本，也是 HTML 4.0 的重新组织。浏览器开发者可以选择两种不同的方式达到不再设计私有标签的目的：将 XML

代码段包含在 XHTML 代码里，或将 XHTML 代码段包含在 XML 代码里。虽然目前使用 XHTML 的网站还不多，但它代表着未来的发展趋势。作为渗透测试人员，应该关注 XHTML 的内容及其发展，从而寻找新的安全漏洞。

5. JavaScript

JavaScript 是一种脚本语言，它最初的名字是 LiveScript。JavaScript 和 Java 没有任何关系，它更不是 Java。Sun Microsystems 创造的 Java 是一种编译语言，Netscape 的 Brendan Eich 于 1995 年创造的 JavaScript 是一种客户端的解释语言。两者之间的唯一关系就是，出于市场营销目的，它们取了相似的名字。

JavaScript 已成为 Web 页面开发者和浏览器生产厂商运用的客户端脚本编程的标准。这个语言支持在 Web 页面上直接与用户交互而不必在客户端和服务器之间交换数据。例如，所有烦人的弹出窗口、警告框、评估电子邮件地址有效性的表单，或许都是某些编写很好的 JavaScript 代码的成果。可以这么说，只有想不到的、没有做不到的，JavaScript 几乎有无限的能力。

6. JScript

很多人都误以为 JScript 就是 JavaScript 的缩写，事实并非如此。下面介绍一下它的来历，在 Web 脚本语言世界中第一个诞生的语言是 NetScape 创造的 JavaScript。到了 1996 年 11 月，ECMA 开始根据 JavaScript1.1 的规范，着手制定 Web 脚本语言的标准，这个标准首次发表于 1997 年 6 月公布的 ECMA-262 号白皮书。此后，所有浏览器厂商才终于有了一个可遵循的规范，让 JavaScript 的市场接受度向前迈了一大步。

拥有平台优势的 Microsoft 原本希望能以 VBScript 与 JavaScript 在客户端平台上一决高下，可惜市场反应并不如预期的那样。因此 Microsoft 很快改变策略，其策略很高明——“如果不能打败它，就迎向它”。Microsoft 的工程师们在最短时间内，根据 ECMA-262 标准制定了 JScript，并且让不断推陈出新的浏览器工具持续支持 JScript 版本的更新。灵活的市场策略让 JScript 颇具生命力，也开始受到网页工程师们的青睐。

7. VBScript

Visual Basic 是一种易学易用的高级语言，它于 1991 年首次发布。从 Visual Basic 中，微软创建了一种简化的脚本语言，称为 VBScript。与 JavaScript 相似，VBScript 学起来简单，并且易于使用，得到了广泛支持。现在网络上有很多专门提供 VBScript 示例的网站，他们提供了 VBScript 的众多应用。VBScript 在微软系统的很多方面得到了应用，如 ASP 页面、客户端 DHTML、Windows、Office 系统编程等。VBScript 存在的问题是，它仅能工作在微软的 Windows 平台上，不能在其他平台上使用。但由于微软产品在市场上的强势地位，这点问题实际上也就显得微不足道了。

8. Perl

Perl(Practical Extraction and Report Language)是一种很古老的脚本语言，它于 1987 年由 Larry Wall 发明。最初的 Web 应用大多是用 Perl 编写的。Perl 很像 C 语言，使用非常灵活，对于文件的操作和处理像 C 语言一样方便快捷。Perl 原名 Pearl，但在官方发表前，拉里·沃

尔发现已经有一个程序语言叫 Pearl，他就将这个程序语言命名为 Perl。Perl 在初期发展十分缓慢，但不久就得到了广泛应用，几乎每个主流操作系统都支持 Perl，包括 Windows、UNIX 和 Linux、IBM 的 MVS、Cray 超级计算机、MacOS、DEC 的 VMS、OS/2、AS/400 等。

与 VBScipt 和 JavaScript 一样，Perl 也是一种解释型语言。但为了隐藏源代码，也可以做一定程度的编译。Perl 功能强大、使用灵活，使用它几乎可以做任何事情，包括服务器端的 CGI 脚本编程、编写如 Whisker 这样的黑客工具、甚至编写独立的应用程序。

### 9. ASP

ASP(Active Server Page，动态服务器端页面)是一种含有 VBScript 或 JScript 脚本程序代码的网页，其网页文件的后缀是.asp。它是服务器端的技术，用于让 Web 服务器根据用户的要求动态生成页面。比如你在搜索引擎中输入了参数“鱼”，那么服务器返回有关鱼的信息的搜索结果页面；如果另一个人在同一个页面中输入了“狗”，那么返回有关狗的搜索结果。这样根据不同的要求动态生成不同 Web 页面的技术就是服务器端的技术，它们的行为都与 ASP 类似。

### 10. CGI

CGI 是通用网关接口(Common Gateway Interface)的缩写，它是第一个在服务器端生成动态页面内容的技术。CGI 并不使用 VBScript 或 JavaScript 来创建页面，而是使用两种更古老一些的语言来创建页面，它们是 Perl 或 C。实际上，CGI 可以使用下述任何语言来创建页面：Any of the UNIX shells，AppleScript，C 或 C++，Fortran，Perl，TCL 或 Visual Basic。

CGI 通常依靠名为 cgi-bin 的目录作为指示符来确定在将内容返回给客户端之前什么时候执行代码。CGI 本身并不是一种真正的编程语言，而是说明如何使用其他语言来创建内容的指南。Perl 是 CGI 最常使用的语言，程序员创建 Perl 代码文件，扩展名为.pl，之后将代码文件放置在 cgi-bin 目录下。当请求某个.pl 文件时，Web 服务器一旦在目录 cgi-bin 下找到这个文件，就开始执行它。

### 11. PHP

PHP 最初称为 Personal Home Page Tools，也称为 Professional Homepages，现在的规范称为 PHP Hypertext Preprocessor，它由 Rasmus Lerdorf 于 1995 年发明。PHP 是一种开放源代码的脚本编程语言，主要用于 Web 服务器的服务器端应用程序编程，用于动态网页设计。PHP 可用于替代微软的 ASP/VBScript/JScript 体系、Sun 微系统公司的 JSP/Java 体系及 CGI/Perl 体系等。它是一种嵌入在 HTML 页面中的脚本语言。

PHP 是目前使用最广泛的服务器端编程语言之一，它通常与 MySQL 数据库和 Apache Web 服务器一起使用，从而打造了一个全部由免费软件组成的 Web 服务器端方案。PHP 文件的扩展名为.php。与 ASP 相似，PHP 也在 HTML 页面中用标记把代码括起来，它使用的开始标记为“<?php”，结束标记为“?>”。

### 12. ColdFusion

ColdFusion 于 1995 年 7 月诞生在 Allaire 公司，它是由当时的天才二兄弟 JJ 所开发，大哥负责前端程序与界面，弟弟负责后端引擎。因为早期使用 Perl 和 CGI 编写网页相当不便，

既不容易学习、也不够直观，更别提复杂又庞大的项目系统了，开发与维护都给程序员带来了极大困难。因此 JJ 兄弟产生了一个想法，试图以最简易而又直观的语言来展现繁杂高深的技术与功能，这就催生了 ColdFusion。2001 年 Allaire 公司被 Macromedia 并购，2005 年 Macromedia 又被 Adobe 并购。Adobe 对外宣称 ColdFusion 发展会更强大，社区会更活跃。

ColdFusion 使用了一种被称为 ColdFussion Markup Language(CFML)的语言，将标记放置在 HTML 页面中，做法与 ASP 与 PHP 类似。当服务器从硬盘上读取.cfm 页面时，服务器从中寻找以符号“<CF”开始的标记，并处理标记中包含的代码。

13. Java

Java 语言最初称为 Oak，是一个旨在解决所有问题的语言。它已经出现了很长时间，但还没有像 Sun 所希望的那样占领全世界。Java 语言由 Sun Microsystems 的 James Gosling 于 1991 年发明，目标定位为一种未来的语言，号称一次编写，处处运行。这个想法很不错，并且在某种程度上它几乎达到了这个目标。但是，Java 代码依然还存在一些兼容性问题，并且还存在一个速度较慢的恶名。

Java 是一种面向对象的语言，在众多商业网站中都得到了应用。Java 语言可以分为两个主要部分：客户端 Java 和服务器端 Java。下面分别予以介绍。

- 客户端 Java

客户端 Java 在客户端运行，其通常做法是：将 Java 编写的代码放置在扩展名为.java 的文件中，之后将这个文件编译为字节码并保存在扩展名为.class 的文件中，最后，在 HTML 页面中使用标记<Applet>引用字节码文件。当客户端浏览器打开这样的页面时，下载并执行 Java 字节码。客户端 Java 的优点之一是采用字节码让人不容易读懂.class 中的代码，从而增加了理解代码的难度。尽管这种通过模糊实现安全性的方法存在一定的优势，但缺点在于，要执行这些代码的客户端必须下载并安装 Java 虚拟机，以便编译和执行 Java 代码。

- 服务器端 Java

服务器端 Java 使用的模型与 ASP、PHP 和 ColdFusion 这样的服务器端系统使用的模型几乎完全相同。当服务器读入扩展名为.jsp 的文件时，该文件被发送给应用服务器并在该服务器上执行 servlet，然后将结果返回给客户端浏览器。当今市场上的应用服务器很多，但占据最大市场份额的 5 个应用服务器为：Macromedia 的 JRun4，BEA 的 WebLogic，Oracle 的 Jdeveloper，Sun 的 Java Web Server 和 IBM 的 WebSphere。由于 Java 程序可以编译为字节码，因此很多程序开发人员和系统管理员会认为 Java 程序很安全。但需要注意的是，使用反编译器依然可将字节码反编译成 Java 源代码。

## 5.4.2 网站结构

在介绍如何攻击 Web 服务器之前，应该先了解一下 Web 流量的交互过程。在浏览器的地址栏中输入网站的地址后，如 http://www.google.com，客户端计算机首先向附近的 DNS 服务器发送一条 DNS 请求。在这个请求中，客户端计算机询问 google.com 站点的 IP 地址。DNS 服务器返回指定网站的 IP 地址，比如 66.249.89.99。接下来，浏览器创建一个套接字，它是 IP 地址与目标端口的组合，具体到 HTTP 中，目标端口号为 80。这样，创建的套接字地址为 66.249.89.80。然后，Web 浏览器向套接字地址 66.249.89.99：80 发送一条 HTTP GET 请求，

位于这个地址的 Web 服务器侦听到这个请求后，根据情况返回下述响应码之一：200 OK，404 Page Not Found，403 Access Denied 和 302 Object Moved。如果响应码为 200，那么所请求的数据被返回给浏览器，并在浏览器中显示。所请求的数据通常采用标记语言进行格式化，如 XML、HTML 和 SGML。

对 Web 服务器可以进行两种类型的攻击：针对 Web 服务器的攻击和基于 Web 认证的攻击。针对 Web 服务器的攻击包括常用服务器的漏洞利用，如 Apache Web 服务器和 Microsoft Internet Information Server(IIS)，攻击目的通常是上传文件或代码、摧毁服务器或获取私有信息。而基于 Web 认证的攻击旨在获取对网站的非授权访问，通常的做法是使用口令暴力破解。

在 Web 攻击中最困难的是保证 Web 通信能够安全进行。但不幸的是，太多开发人员在其编写的代码中遗留了众多可以利用的问题，太多的服务器没有及时地打补丁，这就产生了本不应该产生的脆弱系统。正是这种漫不经心的态度给攻击者提供了众多容易攻击的系统。作为渗透测试人员，就是要测试这种忽视程度，以便保护网站的安全。

### 5.4.3　电子商务架构

提供电子商务服务的公司在网站中采用的架构可以分为两种形式：单服务器架构与分层架构。在小网站中通常采用单服务器架构。在这个架构中，所有的 Web 服务器都在一台物理服务器上。这种架构存在很大的风险，原因在于服务器的一个部件被突破后，整个服务器及其所有服务都被突破了。作为渗透测试人员，或许没有多少机会去测试单服务器的电子商务网站。因为如果一个公司关心其网站的安全性并雇得起渗透测试人员，那么它就很有可能有足够的资本构造多层架构的网站。在分层架构中，Web 服务分散在多台主机上，并且为了提高可用性通常都会采用冗余机制。

在理解了 Web 语言和电子商务架构的工作方式后，就可以开始学习一些漏洞的识别和利用知识了。下面将介绍两个最常用的 Web 服务器：Apache 和 IIS。

#### 1. Apache HTTP 服务器漏洞

Apache HTTP 服务器是在非营利性组织 Apache 软件基金会(ASF)的指导下开发的。根据 2007 年 2 月份的 Netcraft 调查，Apache 服务器的市场份额为 58.70%，Microsoft IIS 的市场份额为 31.09%。Apache 不像 IIS 那样易受攻击，该服务器上的绝大多数漏洞都出现在其 Windows 端口上，但这个端口不像原始的 UNIX/Linux 版本那样流行。

人们一直都在发现新的漏洞，其中绝大多数都与拒绝服务攻击(DoS，Denial of Service)相关。下面给出一些针对 Apache Web 服务器的常见攻击：

- 内存消耗 DoS——攻击者发送一个带有 MIME 头的 HTTP GET 请求，其中包含了能导致服务器崩溃的大量空格字符。
- SSL 无限循环——攻击者通过终止 SSL 并引发子进程进入无限循环状态而发起 DoS 攻击。
- 绕过基本认证——即使攻击者未被授权访问服务器，他也能获取对受限资源的访问。这种情况仅在 Apache2.0.51 中发现，原因在于阻止 Satisfy 命令出现的代码存在一个缺陷，而 Satisfy 命令授权用户使用用户名和口令或客户端 IP 地址访问服务器。
- IPv6 URI 堆溢出——利用 Codenomicon(一家著名的自动软件测试工具厂商)开发的

HTTP 测试工具，恶意黑客能够在 Apache 运行时库发生输入有效性错误时摧毁服务器。

### 2. IIS Web 服务器

微软一直都在大踏步地加强其 Web 平台的安全性。每一个版本的 IIS 都比前一个版本更安全些，当前版本对网站的方方面面都提供了保护。但是，每一个版本也都增加了复杂性，因此也就增加了服务器未打补丁和不安全的机会。

IIS 不仅是一个简单的 Web 服务器，它还提供了许多服务，主要包括：

- FTP 服务
- NNTP 服务
- Internet Printing Protocol(IPP，互联网打印服务)
- SMTP 服务
- BITS(用于 Windows 更新)
- Internet Information Services Manager
- FrontPage 2002 Server Extensions

在 WWW 服务中，提供了下述功能：

- 活动服务器页面(Active Server Page)
- 服务器端包含
- Webdav 发布
- WWW 服务
- Internet 数据连接器(IDC)
- 远程桌面 Web 连接
- 远程维护

与其他所有服务器一样，应该关闭不需要的服务。许多网站被突破的主要原因就是由于这个网站的服务器上开放了多余服务，如远程维护或 IPP。由于这些服务没有被使用，所以通常都使用了默认设置，这些设置为攻击者打开了大门。

针对 IIS 的常用攻击包括：

- Showcode.asp
- 特权执行
- 缓冲区溢出

## 5.4.4　Web 页面欺骗(钓鱼网站)

Web 页面欺骗也称为钓鱼(Phishing)，这是社会工程的一种手段，它正成为恶意黑客收集用户账户信息的流行方法。下面给出了进行 Web 页面欺骗的基本步骤：

步骤 1：使用如 Wget 或 Teleport Pro 这样的网站下载工具下载要伪装的网站；

步骤 2：根据需要修改网站，以便能够从不留意的用户那里收集信息，如信用卡详细资料等；

步骤 3：开通网站，使用与要伪装的原网站类似的域名，如使用名称 http://www.ebys.net 替代原网站 http://www.ebay.com；

步骤 4：找出被攻击者网站驻留主机的 IP 地址，并将这个地址编码为 32 位双字。黑客

可以通过 ping 网站或使用如 NSLookup、dig、host 这样的实用程序得到驻留主机的 IP 地址；

步骤 5：这是一个可选步骤，黑客还可以使用页面名称的十六进制表示来隐藏 Web 页面的名称。比如，假设页面名为 mypage.htm，通过采用十六进制 ASCII 码替代其中的一些字母，就能够隐藏页面的名称。字母 t 的 ASCII 值为 116，其十六进制值为 0x74。这样，黑客就可以把前面提到的页面名称改为 mypage.h%074m，达到隐藏希望用户访问文件类型的目的；

步骤 6：精心制作电子邮件，请求用户访问自己伪造的 Web 站点。这里，代替链接链接到实际网站，默认链接链接到伪造网站上。

### 5.4.5　Cookie 猜测与隐藏字段

很多网站使用 Cookie 来跟踪用户的活动或实现自动登录。Cookie 是一个简单的文本文件，通常包含 ID 值，恶意黑客可修改这个值并猜测一个值来获取账户的非授权访问。

在 Web 页面上可以使用隐藏字段来隐藏信息，通常情况下，隐藏字段包含的信息都是一些关键信息，如用户名和口令。隐藏字段存在的问题是它并不是真正的隐藏，只是在页面上不显示出来而已，只要知道查看它们的方法，就能把隐藏字段找出来。查看的方法是，从浏览器中选择查看源代码的菜单项，或者使用网站抓取程序下载网站，然后脱机查看源代码。通过浏览代码，就可以知道隐藏字段的作用和功能。

### 5.4.6　暴力攻击

绝大多数人想到 Web 攻击时，就会想到摧毁网站的账号。然而，这并不是一件容易的事，其实它与猜测口令一样困难。口令猜测可以手工进行，就是想象别人可能使用的口令，也可以通过软件程序自动地猜测。

软件口令破解工具依赖两种技术：字典攻击和暴力攻击。字典攻击需要一个包含单词的字典文件，这里的单词通常是常用单词和数字的组合，如 password123 等，软件程序使用字典文件猜测网站的口令。暴力攻击需要花费的时间更长一些，原因在于它检查数字、字母及特殊字符的每一种组合。

Web 认证的方式有两种：

1) HTTP 基本认证：基本认证是提供网站访问的一种简单认证方法。口令以明文形式发送给服务器，如果服务器是 Windows 系统，口令通常保存在服务器的安全账户数据库中。Web 开发人员可以很容易地建立基本认证，因此，这种认证方法在一些小型、简单网站上经常看到。

2) HTTP 基于表单的认证：这种形式的认证也以明文形式将用户名和口令发送给服务器。但这个方法不与 SAM 账户数据库链接，而是使用自己的账户数据库(通常是 SQL 数据库)。基于表单的认证要求设计定制的 Web 页面，因此需要做更多的工作。这种认证是大型网站最常采用的方式。

作为渗透测试者，可能会遇到这两种类型的认证。基本认证常应用在网络设备上，如 Cisco Visual Switch Manager(VSM)，该系统运行在 Catalyst 交换机上。基于表单的认证常常在网站认证上，它的账户信息通常保存在数据库中。了解将要攻击的系统所使用的认证类型很重要，因为认证类型指明了破解登录凭据所需要使用的实用程序的类型。破解 Web 口令的两个常用实用程序是 Brutus 和 HTTP Brute Forcer，这两个程序都是由 Munga Bunga 开发的。

### 5.4.7　获取网站信息

Google 是一个寻找脆弱系统及其他信息的优秀工具。在 Google 中搜索常见的出错消息，就能够找到与这个漏洞相关的所有网站。某些登录入口为系统管理员提供了完成各种管理任务的访问点，而从这些访问点处可以获取服务器上在用软件的有关信息，这就为拥有该软件漏洞利用代码的黑客打开了方便之门。因此，本小节将着重介绍服务器上的信息是如何泄露的以及防止这些信息泄露都可采取哪些方法。

#### 1. 查看目录列表

目录列表是一种列出 Web 服务器上文件和目录的 Web 页面。这种页面被设计为通过单击目录链接进行来回导航的形式，它通常包括一个描述当前目录的标题、一组可以单击操作的文件和目录以及通常存在于目录底部的一个注脚。目录列表在分析 Web 服务器上的应用时十分重要，因为其底部的服务器标志显示了正在运行的 Web 服务器软件的细节。

#### 2. Web 服务器软件出错消息

出错消息包含了大量有用信息，有时甚至可使用各种出错消息来寻找运行特定版本软件的服务器。下面介绍一下 Web 服务器本身生成的出错消息。

- Microsoft Internet Information Server(IIS)

寻找出错消息的最佳方法是了解服务器能够生成什么样的消息。收集这些消息可以通过考察服务器源代码、配置文件的方法，也可以通过在服务器上引发错误的方法。但最佳的方法是考察出错页面的源代码。

- Apache Web 服务器

通过搜索服务器生成的出错消息也可以定位 Apache Web 服务器。

#### 3. 应用软件出错消息

前面介绍了 Web 服务器产生的出错消息。很多情况下，运行在 Web 服务器上的应用软件也会产生能够披露服务器信息的出错消息。互联网上有数不清的 Web 应用程序，每一个程序都会产生或多或少的出错消息。诸如 SPI Dynamic WebInspect 这样的专用 Web 评估工具擅长完成详细的 Web 应用程序评估，这就使得利用 Google 搜索应用程序的出错消息显得用处不大。但本书建议搜索出错消息，原因是包含在出错消息中的数据不应该被忽略。前面已经介绍了一些出错消息，后续章节还将介绍更多这方面的内容。虽然本书不可能深入探讨这个主题，但会全力展现 Google 定位这些区分性很强的出错消息的能力。

#### 4. 默认页面与默认文档

定位特定类型服务器或 Web 软件的另一种方法是搜索默认 Web 页面。绝大多数 Web 软件(包括 Web 服务器本身)都带有一个或多个默认或猜测页面，这些页面用于帮助站点管理员测试 Web 服务器或应用程序的安装。通过提供简单的测试页面，管理员能够轻松地把浏览器连接到自己的 Web 服务器上，从而评测 Web 软件安装的正确性。某些操作系统甚至自带了已经安装好的 Web 服务器，而此时机器的主人甚至根本不知道自己的机器上运行着 Web 服务器。这种不常见的行为可让攻击者准确地判断出该 Web 软件没有得到良好地维护，并且不

安全。再进一步推广一下，由于维护很差，攻击者还可以假定该服务器的整个操作系统都存在漏洞。

某些情况下，Google 在 Web 服务器刚刚安装后就抓取其页面，并显示一组默认页面。此时，在 Google 抓取站点内容与站点实际放置内容之间就存在一个窗口期，这就意味着网页的实际页面与 Google 对该网页的快照不一致。这一点对 Google 黑客来说也很有用，因为过去的默认页面也能很好地用于勾勒服务器的特征，但需要提醒的是，本质上所有的搜索内容都是 Google 的缓存页面。不管服务器安装默认页面的原因是什么，但总有黑客对 Google 搜索到的默认页面感兴趣。

尽管 IIS 的每一个版本都会显示不同的默认 Web 页面，但某些情况下，服务包或修补包会修改默认页面的内容。此时，细微的页面变化也可以植入到搜索条件中，这样不仅可以搜索到操作系统版本和 Web 服务器版本，而且可以得到服务包版本与安全补丁版本。这些信息对于那些致力于不仅攻击 Web 服务器，而且要攻击到 Web 服务器之外并进入操作系统本身的攻击者来说简直就是无价之宝。绝大多数情况下，控制了操作系统的攻击者能比仅控制 Web 服务器的攻击者造成更大的破坏。

通常情况下，Web 服务器软件通常都带有安装在 Web 目录下的手册和文档。攻击者也可以利用这些文档来了解服务器的信息或确定运行在上面的 Web 软件的信息。

### 5.4.8 Web 攻击的检测与防止

检测 Web 攻击可以很简单，仅需要浏览一下 Web 服务器的日志中是否存在数十次、甚至数百次的文件和目录访问企图。也可更进一步，将日志文件与功能完善的网络 IDS(比如 Cisco ID 传感器)结合起来，这能为管理员提供更详细的攻击信息。

目录遍历是针对 Windows IIS 4.0 和 5.0 的一种常见攻击方法，它让黑客能够执行或访问 Web 服务器文件夹之外的文件。例如，目录遍历让黑客能够执行 cmd.exe/c 命令来提取目录信息或运行 Web 服务器上任何可用的可执行程序。考虑一下基本攻击，观察日志文件中记录了什么内容。常用的 CGI 漏洞扫描器是 Whisker。与其他自动测试工具一样，该工具通常也是对 Web 服务器执行每一种可能的攻击，从而触发大量的警报和事件来获取有关服务器以及 Web 软件的信息。

防止 Web 攻击是一件费时的任务，从操作系统到网络架构，必须对系统的方方面面都提供保护。下面是几个通常要考虑的保护领域：操作系统，网站设计和网络结构。

#### 1. 操作系统安全防护

Web 服务器的操作系统选择十分重要，Linux 和 FreeBSD 是操作系统中完成基本安装后比较安全的两款操作系统。然而，与 Windows 服务器一样，它们也可能没有进行安全配置就用作 Web 服务器。这样，安全性上将存在较大风险。当今的通用操作系统都有这样的问题，它们默认地运行着大量服务和应用，而完美的 Web 服务器系统应该只有一个应用在其中运行：即 Web 应用本身，如 IIS 或 Apache。

为了保护操作系统的安全性，应该完成下述任务：

1) 修改及重新命名默认账户；

2) 使用较长的口令；

3) 卸载所有未使用的应用程序；
4) 关闭所有不需要的服务；关闭对诸如 cmd.exe 的程序的访问；
5) 使用拥有最低必须权限的用户启动 Web 服务器；
6) 配置允许端口，如 80 和 443；
7) 安装反病毒程序(根据需要选择)；
8) 安装基于主机的 IDS，监控日志文件；
9) 创建一种将日志文件归档到其他位置(不在 Web 服务器上)的机制；
10) 应用所有最新的服务包和更新；
11) 将 Web 页面放置在非标准的目录或驱动器上。

由于每天都会发现新漏洞，因此，保护操作系统的安全就是一个永不停歇的工作。然而，通过研究操作系统厂商 Web 站点公布的保护操作系统安全的最新方法及途径，也可以得到近乎完美的系统安全。

### 2. 保护网站设计的安全性

保护网站设计的安全性是一个很难介绍的题目，因为它包含了开发人员无意中遗留在 Web 中的各种可能的问题。这里介绍的内容仅包括用于创建动态内容的部署技术的实现。例如，ASP 开发人员可能会创建一个仅允许输入 10 个字符的 Web 页面，但他又没有实现用户输入字符串长度的检查，这样黑客就能发送更长的数据，从而可能产生意想不到的后果，如摧毁 Web 服务器等。另外，当开发人员没有检测由输入生成的 SQL 语句时，就在应用程序中留下了 SQL 注入的漏洞，这样可能导致黑客在数据库中添加、删除、修改、查看数据。这些问题及其他常见设计缺陷都会给 Web 服务器安全带来威胁。为了减少安全威胁，Web 设计人员应该重视下述实践：测试 Web 页面上用户输入的有效性，测试客户端返回到服务器的所有数据的有效性以及加密 Cookie 信息。

### 3. 保护网络结构的安全性

应该把所有Web 服务器都放在由防火墙保护的独立安全网络中，以便阻止除允许流量之外的所有流量。在 Web 服务器环境中，典型允许的流量是通过端口 80 和 443 的流量。防火墙可以选择 Cisco PIX Firewall 或 Cisco Adaptive Security Appliance。

## 5.5　数据库攻击

信息是企业和机构的生命线，现在每个成功的企业都在使用数据库。具备了以某种形式存储业务信息的能力，就意味着不用走到仓库实际清点库存就能够回答诸如“现在仓库中还有多少笔记本？”这样的问题。数据库无处不在，每当我们使用搜索引擎、咨询电话号码、到超市购物，实际上我们都在间接地使用甚至更新数据库。用最简单的话说，数据库就是保存数据的一个仓库，它在物理上可以是一个或多个文件，但对用户来说，他所看到的是用二维表表示的、按行和列组织的数据集合。

数据库实际上牵涉了两项：一个数据库管理系统，这是一个软件，它负责数据库的创建、

管理、操作等任务；另一个是存储着用户与系统数据的实际数据库，它是由管理系统创建的。一个数据库管理系统上往往能够创建多个数据库，也能够管理这些数据库。常用的数据库管理系统包括 Oracle、Sybase、Informix、Microsoft SQL Server、MySQL、Microsoft Access、Visual FoxPro、DB2 等。

现在人们使用最多的数据库系统都是关系型数据库系统，它们采用了标准的 SQL 语言作为数据操作和定义语言。虽然每一种数据库的实现都存在一些差异，但基本上都是对标准的扩充。在实际应用中，并不需要用户直接输入 SQL 语句来查询信息，而把 SQL 语句编写在应用程序中，程序在运行过程中，根据需要把 SQL 语句发送给数据库管理系统，数据库管理系统将满足条件的数据或操作结果返回给应用程序。例如，当访问联机销售网站时，在它的价格范围框中输入或选择一个价格范围后，Web 应用就会向数据库发送一条 SQL 语句，查询满足这个条件的商品的信息。不同厂商出品的数据库管理系统在 SQL 语法的采用上存在一些不同，但它们基本上都支持国际标准 ANSI SQL-92，最新标准是 ANSI L-2003。

数据库的设计正朝着易于用户提取数据的方向发展，为方便应用，数据库必须提供某种形式的外部窗口，让程序能够访问数据库中保存的数据。然而不幸的是，这个窗口也给黑客留下了进入数据库、提取数据库信息的大门。由于下述一些原因，数据库经常会受到黑客的攻击：

- 数据窃贼——攻击数据库的一个最显而易见的原因是，攻击者想要得到数据库中包含的数据。例如，信用卡的详细信息无疑要保存在数据库中，这些数据是黑客关注的重点目标。另外，数据库攻击也是工业间谍的一件不可或缺的工作。其他一些数据，像对于竞争对手来说的客户信息，也面临着类似的风险。
- 数据操纵——达到拒绝服务攻击的途径有多条，其中包括删除和修改数据、删除用户账户、甚至完全关闭数据库服务器。
- 系统级利用——数据库也可以成为网络上其他系统的后门。如 SQL Server 这样的数据库都提供了一些例程，黑客不仅可以利用它们攻击和进入数据库，还能够获取服务器的管理权限，从而控制整个网络。

### 5.5.1　常用数据库简介

目前常用的数据库都是关系数据库，也称为关系数据库管理系统(RDBMS)，这些系统不仅提供了存储数据的能力，也提供了管理和操作数据库中数据的工具。这些工具原本是提供给数据库管理员或开发人员使用的，但它们也是黑客攻击工具箱中不可或缺的工具。了解数据库市场上的主要数据库厂商及其产品很重要。理解网站或应用程序使用的数据库类型及模式有助于快速找到它们的弱点。

#### 1. Oracle 数据库

Oracle 数据库由一组数据库管理工具组成，它们用于维护和操作数据库中的数据、定义数据库的结构和管理数据库用户的权限。第一个 Oracle 数据库产品于 1979 年发布，现在已经发展到 Oracle 11g。Oracle 支持多种平台，主要包括 Solaris、Linux 和 Windows。

Oracle 数据库中的数据，逻辑上存储在表空间中，物理上保存在数据文件中。表空间又进一步划分为一些段，如数据段和索引段，目的是将不同区域用于存储不同的内容，达到提

高效率的目标。为了跟踪和记录数据的存储情况，Oracle 使用了一个称为系统表空间的表空间。这个表空间中存储了数据字典及其他一些系统信息。数据字典则是一组包含了数据库中所有用户对象信息的表。

Oracle 提供的 SQL 语言称为 PL/SQL(Procedural Language/Structured Query Language)，它在 SQL 标准的基础上增加了一些 Oracle 专用的东西。用户可以在命令行界面中使用 SQL 语句访问数据库，也可以用如 Oracle SQL*Plus 这样的图形用户界面执行 SQL 语句。此外，还可以使用第三方厂商的工具操作 Oracle 数据库，如 Toad for Oracle，这些工具的特点是方便、快捷。

#### 2. MySQL 数据库

MySQL 是由 MySQL AB 发布和支持的关系数据库管理系统，它是开源的，深受编程人员的喜爱。MySQL 是一个流行的 Web 应用数据库，通常和 PHP 结合使用。它支持多种平台，包括 Solaris、Linux 和 Windows。MySQL 的早期版本不支持真正的 RDBMS 所具备的很多标准功能，包括不支持事务处理。现在这种情况已经被改变，从版本 5.0 开始提供了对存储过程和视图的支持。

与其他常用关系数据库管理系统一样，MySQL 数据库逻辑上由表组成，表包含在表空间中，它在物理上存储为数据文件。每一个 MySQL 数据库都映射为 MySQL 数据目录下的一个目录，数据库中的每一个表都映射为数据库目录下的一个文件。从安全上看，MySQL 易受到攻击，原因在于读取存储在这些文件中的数据相对要容易些。另外，从 5.0.2 版开始，通过查询一组 INFORMATION_SCHEMA 视图，能从 MySQL 中提取到源数据，而这些视图也是基于保存在 MySQL 数据库中的数据得到的。MySQL 在实现 SQL 时对标准提供了变通支持，并包括了一个启动 MySQL 服务器时用于选择 ANSI 模式的开关。MySQL 在版本演化过程中变化较大，与 ANSI 的兼容性的变化也很大。如触发器这样的功能，它仅从 5.0 版才开始提供基本的支持，在此版本之前，它不支持存储过程。

查询 MySQL 有多种，主要包括：MySQL 命令行工具；MySQL 控制中心(mysqlcc)，这是一个独立于平台的图形用户界面工具；MySQL 查询浏览器(MySQL Query Browser)，这是 mysqlcc 的一个更新版本。

#### 3. SQL Server 数据库

SQL Server 是微软提供的关系数据库管理系统，它的第一版于 1989 年发行，由于它是微软的产品，因此只支持 Windows 平台。这个数据库的最新版本为 SQL Server 2005，按照微软的官方说法，它是用于大规模联机事务处理(OLTP)、数据仓库和电子商务应用的数据库平台，也是用于数据集成、分析和报表解决方案的商业智能平台。

SQL Server 数据库的数据逻辑上表示为数据库表，物理上存储在一个或多个操作系统数据文件中。SQL Server 使用了 4 个系统数据库，这些数据库在创建数据库服务器时生成，并且是 SQL Server 服务器正常运行不可或缺的基础。

SQL Server 使用 Transact-SQL(T-SQL)作为它的查询语言，T-SQL 是标准 SQL 的一种变体，可以直接在命令行工具 osql 中运行 SQL 查询，也可以使用它的图形界面工具“企业管理器”来提取数据和执行 SQL。

### 5.5.2　攻击 SQL Server 数据库

本节介绍数据库系统的一些潜在攻击点，主要精力集中在 SQL Server 数据库上，但它们包含的思想绝大多数能运用到所有关系数据库管理系统上。在攻击数据库系统时，DBA 的工具箱通常就是黑客的工具箱，SQL Server 的两个标准工具(查询分析器和命令行工具 osql)也是数据库黑客工具箱中的两个常用工具。

#### 1. SQL 注入

Web 应用通常都会使用数据库，为用户提供读、写数据的能力。但大量Web应用都存在 SQL 注入(SQL Injection)漏洞。简单地说，SQL 注入是指攻击者通过修改 Web 应用要发送给数据库的 SQL 语句，达到获得某些他想得知的数据(或完成某个操作)的目的。如果想要深入理解这个概念，则需要了解一些关于 SQL 的知识。

#### 2. 系统存储过程

存储过程由一条或多条 SQL 语句组成，它通过过程名称进行调用。绝大多数关系数据库管理系统都支持存储过程。一般来说，程序员会将几条 SQL 语句封装到存储过程中，完成一个完整的逻辑过程。

SQL Server 提供了很多内置的存储过程，称为系统存储过程和扩展存储过程。这些存储过程在数据库服务器的标准安装中默认地进行安装。尽管绝大多数存储过程都是数据库正常运行所需的工具，但其中的一些存储过程能被黑客利用，从而对数据库造成危险。许多系统存储过程对于数据库应用开发都十分有用，但在生产环境中许多系统都没有去除这些存储过程。曾经有一段时间，人们建议把最危险的存储过程彻底地从系统中删除，但这种做法会对数据库系统的其他方面造成重大冲击。因此，现实的做法是，通过严格控制存储过程的使用权限来最大限度低降低它们所带来的安全风险。

#### 3. 连接字符串与口令破解

为在 Web 应用中连接数据库，很多 Web 开发者采用硬编码方式将连接字符串定义在配置文件中，如 web.config 或 global.asa。

SQL Server 将用户名和口令存储在 master 数据库的 sysxlogins 表中。口令使用名为 pwdencrypt()的存储过程序列。然而不幸的是，SQL Server 不仅存储了口令的散列结果，而且是对口令进行大写归一化后散列，因此减少了口令的可用空间，使得口令破解变得更加简单。

现在有各种各样用于暴力破解 SQL Server 口令的工具，包括：

- SQLAT——以暴力攻击模式或字典攻击模式运行的口令审计工具。
- SQLDict——字典攻击工具。

此外，下面的存储过程也能完成类似任务：

- FindSA——用于得到 SA 口令的暴力破解工具；
- FindSADic——用于得到 SA 口令的字典攻击工具。

### 5.5.3　攻击 Oracle 数据库

攻击 Oracle 数据库服务器的第一步工作是找到 Oracle 服务器，在知道它在什么机器上运

行后，TCP 端口扫描是找到 Oracle 数据库服务器的有效途径。Oracle 及其辅助进程侦听众多的不同端口，虽然这些端口可以设置，但多数情况下人们会全部或部分使用默认端口。

PL/SQL 是Oracle 版本的 SQL，它用于创建存储过程、函数、触发器及对象。PL/SQL 是 Procedural Language/SQL 的缩写，也是操作 Oracle 数据库的主要工具。对于专门从事数据库攻击的人来讲，应该阅读一本详细介绍 PL/SQL 的图书。

PL/SQL 注入是一种与 Oracle 存储过程相关的重要攻击技术。利用 PL/SQL 注入，攻击者能够提升其权限，从较低级的 PUBLIC 账户提升为具有 DBA 权限的账户。这项技术适用于几乎所有 Oracle 版本，能用于攻击定义存储过程和 Oracle 本身自带的存储过程。

### 5.5.4　保护 SQL Server 的安全

为了方便开发和使用数据库，SQL Server 在默认安装中提供了一个便利的开发环境，它提供了丰富的特性，包括示例数据库、完整的存储过程及其他用于方便管理和使用数据库的特性。但是，这些便利的开发环境工具在生产环境中却可能会带来众多不安全因素，成为网络的后门。因此，必须在 SQL Server 投入生产环境之前，实现一组标准的安全防护，从而让生产环境变得安全可靠起来。

#### 1. 用户认证

与其他绝大多数关系数据库管理系统一样，SQL Server 能够管理其自身的安全并在内部维护其用户和口令，它所使用的数据库表为 sysxlogins。SQL Server 支持两种认证方式：本地的 SQL Server 认证(称为混合模式)和 Windows 认证。用户可在安装 SQL Server 时配置认证模式，也能够在安装后修改认证模式。

Windows 认证使用 Windows 集成安全自动认证要登录到数据库的用户，它利用了用户登录域时所用的凭据。混合模式认证既允许用户使用自己的 Windows 凭据登录数据库，也允许用户使用基于 SQL Server 的安全登录数据库。不论使用哪一种登录方式，都必须在数据库级创建登录用户。如果采用了 SQL Server 登录，那么口令以单向散列方式存储在 sysxlogins 表中。

其实，使用 Windows 认证模式本质上更安全，原因在于口令管理的完整性，如口令最短长度、多次无效登录后的账户锁定等。如果认证模式从混合模式改为 Windows 模式，那么已有的任何 SQL Server 登录用户都不再有效，不能再通过认证登录数据库服务器。

#### 2. 服务账户

在标准 SQL Server 安装过程中，默认地安装了许多服务，使用 SQL Server 正常发挥功能必须运行的服务是 MSSQLServer 服务。但 SQL Server 调度代理 SQLServerAgent 通常在标准安装后都会被安装和运行。除了这两个服务之外，默认安装还会安装下述服务：MSSQLServerADHelper、MSDTC(Distributed Transaction Coordinator)和 MSSearch Services。

关闭未使用的服务有助于缩小黑客攻击面，这是一种良好的实践策略。如果选择了默认安装，这些服务都将被安装，并运行在本地系统账户中。本地系统账号在本地机器上拥有一些权限，包括两种形式：以操作系统方式操作(Act as Part of the Operating System)和生成安全审核(Generate Security Audits)。用户应该让服务器运行在不必是本地管理用户组或域管理用

户组成员的域用户账户下，并且确保运行服务的账户拥有能够让服务正常运行的最小权限。

#### 3. Public 角色

与 Windows 一样，每个 SQL Server 安装都有一些预先定义的角色。这些不同的角色能够影响整个服务器的权限级别或单个数据库的权限级别，其中最重要的一个角色是 Public 角色，它被授予数据库级别的权限。每一个数据库都拥有 Public 角色，添加到数据库中的每一个用户都是 Public 角色的一个成员。这种约定无法改变，用户既不能删除 Public 角色，也不能从这个角色中删除成员。因此，必须小心谨慎地管理 Public 角色的权限。

#### 4. Guest 账户

SQL Server 中有一个称为 Guest 的账户。如果这个 Guest 账户是数据库级账户，那么任何登录都能够访问数据库。由于 Guest 账户是 Public 角色的成员，通过这个账户访问数据库的任何用户都拥有 Public 角色拥有的权限，而无须经过授权。为了降低 Guest 账户引发的风险，就要确保 model 数据库中不包含这个账户，因为 model 数据库是新创建所有用户数据库时使用的模板。此外，需要定期审计数据库用户，检查是否创建了 Guest 账户。

#### 5. 网络库

SQL Server 支持集中网络层协议与客户端通信。这些协议包括 TCP/IP、IPX/SPX、Named Pipes 和 AppleTalk。为完成这些通信，SQL Server 使用称为网络库(或 Net-Libraries)的动态链接库来进行通信，要求在服务器和客户端都安装这些网络库。如果客户端和服务器端的配置支持相同网络库，客户端发起连接，则只能与 SQL Server 进行通信。需要注意的是，不要混淆网络协议和网络库，虽然它们的名字很接近，但一般来说，一个网络库可以支持多个网络协议，SQL Server 的默认安装使用了 TCP/IP 和 Named Pipes 网络协议。

虽然黑客不能攻击 SQL Server 的网络库，但网络库表明了客户端用于通信的协议。开发超过需要的网络库只会给系统带来更大的风险。一般来说，只要有可能，就应该使用 TCP/IP 网络库，原因在于它有较快的速度和较好的安全性。另外，使用 TCP/IP 网络库还可以修改 SQL Server 的默认端口(默认端口 1433)，选择选项“隐藏服务器(Hide server)”后，可将默认端口切换为 TCP 2433，但微软不建议修改默认端口。

### 5.5.5　保护 Oracle 的安全

保护 Oracle 的安全是十分艰巨的任务。原因很简单，Oracle RDBMS 实在太庞大了。本节将介绍一些有助于保护 Oracle 环境安全的简单措施。

#### 1) 设置 TNS Listener 口令与打开 Admin Restrictions

默认情况下，TNS Listener 没有设置口令，它能被任何人员通过远程连接来维护(这种做法在 Oracle 10g 中已经被修改)。设置 Listener 的口令将能有效地阻止对 Listener 的非授权管理。

通过打开 Admin Restrictions，将阻止对 Listener 的非授权维护。当 Admin Restrictions 打开之后，即使提供了 Listener 的口令，某些命令也不能远程执行。

#### 2) 打开 TCP 有效节点检查

TCP 有效节点检查用于允许某些主机连接到数据库服务器，而阻止另一些主机连接到数据库，从而将脆弱数据库的主机限定在某个已知范围内。

#### 3) 关闭 XML 数据库与关闭外部过程

XML 数据库(XDB)提供了两项服务：一个是侦听 TCP2100 端口的 FTP 服务，另一个是侦听 TCP8080 端口的 HTTP 服务。如果我们没有使用这些服务，那么就应该关闭它。

外部过程让 PL/SQL 过程能够调用操作系统共享对象(比如 DLL)的功能。这种调用存在潜在的威胁，如果不需要这种功能，就应该及时关闭它。编写 PL/SQL 代码的开发人员应该尽可能地避免使用外部过程。

#### 4) 加密网络通信与锁定和过期未使用账户

加密网络通信设置仅仅适用于 Oracle Enterprise Edition，在这个版本中，应该使用 Oracle Advanced Security 加密客户端与数据库服务器之间的流量，使用 Oracle Net Manager 工具设置这个选项。

侵入 Oracle 服务器最容易的方法或许是得到它的用户名和口令。因此，所有未使用账户都应该锁定或使其过期，可使用数据库配置助手(DCA，Database Configuration Assistant)来完成这个任务。

## 5.5.6　检测数据库攻击

数据库攻击经常发生在代码编写得糟糕的情况下。下面介绍检测数据库攻击的方法。

### 1. 审计与失败登录

审计通常是数据库管理员识别数据库的非授权访问或访问企图的第一个步骤。访问审计比较容易设置，可在服务器级或数据库级配置访问审计。除了 Windows 事件日志外，SQL Server 提供了自己的出错日志记录系统。当在 SQL Server 中配置访问审计时，审计事件同时写入到两个日志中。SQL Server 提供了 4 个级别的审计：无(None)——不记录登录尝试；成功(Success)——只有成功登录被写入到日志；失败(Failure)——只有失败登录被写入到日志；所有(All)——每一个登录数据库的尝试都被记录到日志中。作为进一步提高保护层次的一种手段，也可以使用入侵检测系统(IDS)来阻止非授权访问。

与其他应用程序一样，SQL Server 的口令也能够被暴力破解。前面已经介绍过一些用于暴力破解 SQL Server 口令的工具。在打开失败登录监控后，当发生暴力破解 SQL Server 口令的现象时，日志中就会产生大量失败登录记录。

### 2. 系统存储过程与 SQL 注入

SQL Server 提供了众多系统存储过程，sp_password 便是用于添加或修改 SQL Server 登录的口令。这个存储过程在应用中有一个特性：每当调用这个存储过程时，系统就会识别出名称 ps_password，因此整条语句都从工具中隐藏起来。

SQL 注入的本质使得检测这种攻击几乎不可能，不管这种检测是实时检测还是其他某种类型的审计检测，即使是 IDS 也几乎检测不到 SQL 注入。防止 SQL 注入攻击的唯一有效手

段是实现全面的代码评审并在生产系统中严格遵从编码标准。确保整个应用程序中都进行输入有效性认证，对输入数据进行检查，删除任何具有潜在危险的字符，如单引号(')和分号(;)。

### 5.5.7　防止数据库攻击

防止数据库攻击需要做多方面的工作，有些工作很简单，而另一些工作可能会带来庞大的维护任务，从而只能在一些高风险环境中应用。下面是一些防止数据库攻击的预防性措施：

- 服务包和补丁：及时更新操作系统和数据库应用的所有服务包和修补包。
- 物理安全：数据库服务器应该放在一个物理安全的环境中，因为潜在的黑客既可以在单位之外，也可以在单位之内。
- 防火墙规则：使用防火墙阻塞 UDP 端口 1434，这个端口是蠕虫病毒 Slammer 的攻入点，SQL Server 的正常工作不要求开放这个端口。
- 关闭不使用的特性：如果系统中不需要使用 SQL Mail 或其他一些特性，那就关闭这些特性。如果没有调度或报警需求，那么 SQL Agent 就不需要运行。当关闭了这些特性后，就减少了服务器受到攻击的机会。
- 分析工具：可使用各种工具来寻找系统中存在的潜在漏洞。微软的安全审计工具 Microsoft Baseline Security Analyzer(MBSA)可以报告服务器上存在的安全问题。
- 审计共享文件夹的权限：确保没有为了方便文件访问而共享了 NTFS 文件夹。如果将 SQL Server 安装在 Windows 域账户下，那么该账户被自动授予访问 SQL Server 需要使用的所有文件夹。如果随后改变了 SQL Server 的运行用户，而这些权限又没有被收回的话，就会给这个用户遗留下不必要的访问权限。
- 服务账户和 sa 账户：这是两个最经常被攻击的账户，应该为它们设置复杂的口令。如果使用混合模式的话，每一个账户都要使用复杂口令。SQL Server 存储口令的方法并不十分安全，此外，它也不能强制实施口令规则。
- Guest 账户：除了 Master 和 Tempdb 外，从所有数据库删除这个账户，因为 Master 和Tempdb 数据库需要使用这个账户来正常地工作。在工作数据库中如果要保留 Guest 账户，那么将它的权限设置得尽可能地低些。
- 输入有效性评估：不要允许互联网应用随意地查询数据库，取而代之的是，使用存储过程访问数据库资源。
- 数据库和 IIS 服务器：应将数据库服务器和 IIS 服务器放在不同应用平台上，并在数据库服务前面加装一个防火墙，阻止外部用户对数据库的访问。这样就对数据库的安全增加了一道防护屏障。
- Windows 域：只要有可能，就使用 Windows 域，这样可以运用集成安全模式，它比 SQL Server 的安全方式有很大提高。
- 信任连接：例如，当使用命令行工具查询工具 osql 时，使用-E 选项，避免在批处理文件书写易于被人看到的硬编码口令。
- SQL Server 服务账户：在 Windows 域中，应该使用域用户。它不需要是本地系统账户或管理员组成的成员，绝大多数情况下，也不需要是本地管理员。采用最低权限策略确保了即使服务器被攻入，黑客能执行服务账户上下文中的命令，它也依然受到该账户权限的约束。

- Model 数据库：由于Model 数据库提供了新创建的所有数据库的模板，因此它也应该内建安全模板，这样无论谁创建新的数据库，系统中创建的每个数据库都拥有了最大程度的安全性。
- Public 角色：定期审计权限。由于每个数据库用户都自动成为 Public 角色的成员，因此严格控制这个角色的权限十分必要。
- 扩展存储过程：对所有潜在易受攻击的系统存储过程都严格地控制其权限，并定期审计它们的权限。这样可以避免无意义的权限修改或粗心开发人员的权限修改带来的安全风险。
- 网络库：只开放系统所需要的网络库，减少 SQL Server 受到攻击的接触面。
- SSL(安全套接层)：只要有可能，就加密客户端与 Web 服务器之间的通信数据。

## 5.6　口令破解

在计算机发明之前，企业和政府机构使用门锁和文件柜来保护数据的安全，而且这种物理安全让人感觉很放心。现在，通过网络能够在任何地方访问企业的数据，物理安全已经不能对数据提供足够的保护了。这样，企业和政府机构就开始转向采用访问控制来保护自己的数据。强健的访问控制至少应该采用下述方法中的两种方法：

- 被认证者知道的东西：个人身份号码或口令；
- 被认证者拥有的东西：SecureID 卡；
- 被认证者是谁：生物特征；
- 被认证者做的事情：在我们签名时监视笔尖压力的变化。

双因素安全是指至少采用上述两种控制方式的访问控制，如通过指纹和口令进行访问授权。四因素认证则包含了所有四个认证方面。然而，绝大多数应用依然使用最弱的安全认证方式——仅使用口令的单因素认证。由于同时使用了用户名和口令，人们往往认为这就是使用了双因素认证，但用户名和口令方式的认证仍是一种单因素认证，因为它仅仅基于人们所知道的东西。

口令提供了最弱的安全认证方式，原因在于人们可以通过破解猜测出口令，而作为渗透测试人员，也应该知道口令破解的方法。人们之所以雇佣渗透测试人员进行口令破解经常是出于下述两个理由之一：策略审计和口令恢复。

当出于策略审计目的进行口令破解时，测试人员试图确定企业是否强制实施了其口令策略。比如，假定企业安全策略中有一条，要求口令必须是 8 个字符长，并使用字母数字的组合，那么口令 b62ki9d6 就是这个策略下的一个强口令。如果对企业的口令测试中发现只有 80%的口令符合企业设定的策略，那么渗透测试人员就可以告知企业这样的策略没有在企业中得到彻底贯彻。

当公司的系统管理员离开企业且没有任何人知道管理员账户的口令时，企业就可能雇佣渗透测试人员来破解管理员的口令。口令破解工具有很多，本节首先介绍口令在服务器中的存储方式，之后概要描述一些最常用的口令破解工具，最后说明一些避免口令被恶意破解的技巧。本节的主要内容是介绍破解现有口令的方法，但若能直接地物理访问计算机，也可以

使用一些工具覆盖或擦除口令。一个很好的工具是 Passware Kit，它除了能够恢复众多常用软件的口令之外，也能够重置 Windows2003/2000/NT 服务器及 Windows Vista/XP/2000/NT 工作站的口令。这个工具的使用很简单，首先利用它生成一张可启动光盘或软盘，然后重启计算机，按照提示选择操作即可。

### 5.6.1　口令散列方法

口令通过只允许知道口令的人访问系统的方式来保护系统的安全。一般情况下，用户需要通过认证和授权两个阶段对系统进行访问。其中，第一阶段确定用户身份，第二阶段确定用户的操作权限。在这两个阶段中，以明文或密文形式将口令发送给系统都是可以的，但由于明文发送方式的安全性较低，所以常采用密文形式发送口令，而最常用的加密方法就是散列法。

#### 1. 使用加密

在介绍使用不同工具破解口令之前，需要理解口令是如何加密的。由于 Windows 和 UNIX 是市场上最流行的两个服务器，因此这里的讨论仅局限于它们两个。这两个系统的主要差别是：UNIX 及其各种变体(包括 Linux)在加密口令时使用了加盐(加密前在口令中添加的一个随机字符串)，而 Microsoft Windows 服务器则没有加盐，这就是 Windows 口令相对更容易破解的原因之一。

#### 2. 微软口令散列

Windows 散列和 LNMAN 散列是微软推出的两种口令散列。

Windows 散列的第一步是将口令转换成 Unicode 编码。Unicode 是一种用唯一的数字表示每一个字符的编码方法，适用于各种语言和平台。这种编码提供了字符的一致编码手段，开发人员可使用 Unicode 编码在一种语言中编写程序或页面，而在另一种语言中轻易地阅读它们。比如，大写字母 A 的 Unicode 编码为 0041。在第一步完成后，用 MD4 算法散列 Unicode 字符串形式的口令。MD4 算法的安全性比 MD5 算法要弱，但它比微软用于加密口令的 LANMAN 散列算法要强一些。

LANMAN 散列算法从 NT3.5 引入，主要用于向后兼容性。当口令长度不足 14 个字符时，它在口令后面添加上一些零，使其补足到 14 个字符，之后口令被转换为大字字母，并划分为两个 7 字符组。如果口令长度不超过 7 个字符，那么第二个字符组中的所有字符都是 0，在运行加密算法后，所有都是 0 的字符串转换为 0Xaad3B435B51404EE，当看到 LANMAN 口令中出现这样的字符串时，就可以知道它对应的口令长度不超过 7 个字符。接下来从两个字符组中分别计算得到一个 8 位字节奇校验的 DES 密钥，将这两个值结合在一起，得到一个 16 字节的单向散列值。由于在破解 LANMAN 口令时一次只需要破解 7 个字符，因此这种口令易于破解。比如，如果口令为 password123，那么破解口令时只需破解出 passwor 和 d123 即可，这比破解出字符串 password123 容易得多。口令中包含的字符个数越少，破解速度也就越快。

Windows 口令存储在安全账户管理数据库(SAM, Security Accounts Manager)中，其文件为 Windows 目录下的\system32\config\SAM。在 Windows 运行过程中，这个文件(SAM)被

锁定，但该文件的一个备份保存在 Windows 目录下的\repair 目录中，并在每次运行应急修复实用程序(ERD)时更新。Windows 口令破解实用程序要求使用 SAM 数据库。

#### 3. UNIX 口令散列

UNIX 口令比 Windows 口令要更安全一些。在 UNIX 系统中，加密口令时使用了加盐操作来生成一个随机值。UNIX 口令使用 DES 算法对口令进行 25 次循环加密，其中首次加密是使用全 0 的 64 密钥，然后将输出值加盐后作为后续 24 次加密的输入。

在所有类 UNIX 系统中，加密后的口令通常存储在/etc/shadow 文件中，更旧一些版本的系统将加密后的口令存储在/etc/passwd 文件中。

### 5.6.2 Web 口令破解技术

认证是一个验证身份的过程。常用的认证手段是使用用户名和口令，此外，也可以使用能够起到认证作用的任何手段，包括智能卡、虹膜扫描、语音识别及指纹等。Web 认证中，常用的认证方法包括：基本认证、数字认证、NTLM 认证、证书认证、表单认证。下面介绍这些认证方法及实施攻击的手段。

#### 1. 基本认证

基本认证(Basic Authentication)是最简单的认证方法，也是很长时间以来应用最广泛的认证方法。基本认证模式的模型是：客户端必须使用用户标识和口令向服务器说明自己的身份。

只有在服务器确认了用户标识和口令对于所请求 URL 的指定保护空间有效时，它才会提供所需的服务。这种认证没有可选认证参数，为了得到授权，客户端发送凭据，凭据由用户标识和口令组成，中间使用冒号(:)分隔，之后使用 BASE64 进行编码。

基本认证具有下述特点：

1) 基本认证是 Web 应用的最基本认证形式；

2) 认证凭据以 BASE64 编码形式进行编码，以明文形式进行传输，易于窃听、易于受到重放攻击；

3) 使用 128 位 SSL 加密可挫败攻击。

当某个特定资源使用基本认证进行保护时，为了通知客户端必须提供用户凭据，HTTP 服务器发送一个 401 认证请求头，客户端在收到这个 401 响应头时，如果客户端的浏览器支持基本认证，它就提示用户输入要发送给服务器的用户名和口令。为得到资源，从服务器上请求的每一个资源都必须提供认证凭据。由于 HTTP 协议是一种无状态协议，因此每一个请求都以相同方式处理，即使这些请求来自同一个客户端。客户端浏览器缓冲所提供的用户名和口令，并与认证域(Authentication Realm)存储在一起，这样如果从相同域中请求其他资源时，能把相同的用户名和口令提供给服务器来认证请求，而不要求用户重复输入它们。这种方式中，浏览器能区分不同站点的私有认证域。

对于严格的安全定义来说，基本认证不应该被认为是安全的。尽管口令以加密形式存储在服务器上，但它以明文形式通过网络从客户端传输到服务器上。使用任何一款包嗅探器就能够捕获明文传输的用户名和口令。另外，每一个访问请求都要传递用户名和口令，而不仅是在第一次时这样做。因此，包嗅探器不必侦听特定的时刻，只要侦听的时间足够长，就能

获得用户名和口令。此外，数据本身也以明文形式在网络上传送，如果网站中包含了敏感信息，那么包嗅探器同样能捕获这些信息，从而造成敏感信息泄露。

### 2. 数字散列认证

数字散列认证(Digest Authentication)通过用户口令 MD5 摘要的形式实现口令的传输。

数字散列认证具有下述特点：

1) 数字散列认证基于质询/响应认证模型；
2) 用户发出不带认证凭据的请求，Web 服务器回应一个指示凭据的 WWW 认证头；
3) 取代发送用户名和口令，服务器使用一个随机值询问客户端；
4) 客户端使用用户名/口令的消息摘要应答服务器。

与基本访问认证相似，数字散列认证模式基于简单的质询/响应模型。在可选头中服务器指定用于创建检查或摘要的算法，默认时使用 MD5 算法。数字散列认证模式使用一个唯一值作为质询，有效回应中包含了用户名、口令、给定唯一值、HTTP 方法、所请求 URL 的检查和默认时的 MD5 检验和。在这种方式中，从不会以明文形式发送口令。

与基本认证模式相同，数字散列认证预先必须定义用户名和口令。这是一个基于口令的系统，面临着所有基于口令系统相同的问题。特别是，这种协议没有定义如何在建立口令时保护它的安全。

实现中可选择不接受以前使用过的唯一值或以前使用过的摘要，目的是阻止重放攻击。实现中也可以选择使用一次一个唯一值或一次一个摘要的方式。这个唯一值对于客户端来说是透明的，它由服务器指定，客户端只是不加修改地把它返回给服务器。尽管这种认证模式中没有实际发送口令，但发送了它的消息摘要，黑客能利用生成的消息摘要来获取对内容的访问，原因在于口令摘要是访问这样的网站唯一所需的信息。

### 3. NTLM 认证

NTLM(NTLanMan)是一个基于质询/响应模式进行用户和计算机认证的微软专用协议。

NTLM 具有下述特点：

1) NTLM 认证是微软专用的 NT LAN Manager 认证，它只适用于 Microsoft Internet Explorer；
2) 集成的 Windows 认证的工作方式与消息摘要认证相同。

在认证过程开始时，用户的系统(客户端)向 telnet 服务器发送一条登录请求，服务器使用随机生成的“令牌”(质询值)应答客户端。客户端使用质询值对当前登录用户加密保护的口令进行散列化，并把散列结果发送给服务器。服务器收到质询散列的响应之后，将它与已知的正确响应相比较，如果两者匹配，那么用户认证成功。但是，这种做法并不安全，因为 NTLM 散列值能够被暴力破解。

### 4. 数字证书认证

数字证书是一个电子文档，它包含了身份信息、公钥并使用认证中心私钥签署的数字签名。希望发送解密消息的个体向认证中心(CA，Certificate Authority)申请数字证书。认证中心是一个签署证书的权威机构，它能担保证书持有人的身份。CA 发行的加密数字证书中包含了申请人的公钥及其他各种身份信息，它通过印刷媒体或互联网公布自己的公钥。

数字证书认证具有下述特点：

1) 数字证书认证比其他形式的认证更健壮；

2) 数字证书认证使用了公开密钥密码学，数字证书能够存储在智能卡上，从而进一步提高安全性；

3) 目前尚不知道存在针对 PKI 安全的攻击。

当用户连接到服务器上进行认证时，他向服务器提供自己的数字证书，证书中包含了公钥和 CA 签名。服务器首先验证证书上的签名是否有效、是否由信任 CA 生成。通过后，服务器使用公开密钥加密算法认证用户，证实其确实拥有与该证书相关联的私钥。加密消息的接收方使用 CA 的公钥解密附着在消息上的数字证书，验证它的真伪。通过后，得到包含在证书中的发送方的公钥和身份信息。利用这些信息，接收方可发送一个加密应答。数字证书最广泛使用的标准是 X.509。数字证书应经得到了广泛的使用。

下面是在认证中使用数字证书的一些示例：

1) 电子邮件：很多人使用数字证书来加密电子邮件(提供机密性)或签署电子邮件(证实身份)。

2) 网络安全：很多机构(包括一些网上银行)都部署了使用数字证书的智能卡或其他安全技术，以此作为提高他们的计算机网络安全性的一种途径。

#### 5. 表单认证

通常情况下，Web 应用程序用来认证自己用户的凭据是 Web 表单，它同时也决定着用户的授权级别。表单认证具有下述特点：

- 这是一种高度可定制的认证机制，它使用 HTML 中的<FORM>和<INPUT>标记构造表单，形成供用户输入其用户名/口令的字段。
- 在用户输入了用户名/口令，通过 HTTP 或 SSL 发送给服务器之后，服务器端的业务逻辑评估用户名/口令的有效性，如果凭据有效，服务器就像客户端发送一个 Cookie，以便随后的页面访问。
- 表单认证技术是互联网上流行的认证技术。
- 客户端生成访问保护资源(比如交易细节)的请求。
- IIS(Internet Information Server)收到请求。如果请求的客户端得到了 IIS 的认证，那么用户/客户端进入到 Web 应用中。如果打开了匿名访问支持，客户端默认地进入到 Web 应用中。否则，Windows 提示用户输入访问服务器资源的凭证。
- 如果客户端不包括有效的认证凭据 Cookie，Web 应用将用户带到提示用户输入凭证的页面。
- 在提供所需凭证后，用户得到 Web 应用的认证。Web 应用确定了请求的授权级别。如果客户端被授予访问安全资源的权限，认证凭据最终被分配给客户端。如果认证失败，将向客户端返回拒绝访问的消息。

### 5.6.3 口令破解工具

了解口令的加密方式后，下面考察一些用于破解口令的工具，这些工具所遵循的攻击策略不外乎字典攻击、暴力攻击以及混合攻击。

### 1. L0phtcrack

如果不喜欢使用命令行工具破解口令，那么可以使用 L0phtcrack，它也称为 LC5，是一个用于破解 Windows 口令的图形用户界面工具。L0phtcrack 曾是市场上最著名的口令破解工具，它最初由@Stake 公司开发，于 2004 年被 Symantec 公司收购，从 2006 开始，Symantec 不再发行 LC5，但依然可从网络上找到 LC5 安装程序，这是一个 15 天的试用版本，但从网上可以找到一个该工具的序列号生成器，从而更长时间地应用这个工具。由于 LC5 已经不再得到开发支持，也可改用 Cain and Abel、John the Ripper 或 Ophcrack 进行口令破解。

利用 L0phtcrack，能够完成下述任务：

1) 破解本地计算机的口令；

2) 破解远程计算机的口令；

3) 通过使用 NT4.0ERD 破解口令；

4) 通过嗅探网络破解口令。

L0phtcrack 能够进行字典攻击、混合攻击和暴力攻击，它还能够评价口令的难度，这将有助于编写渗透测试报告。

### 2. Nutcracker

Linux 和其他 UNIX 变种在口令加密过程中使用了加盐，目的是为了让口令难以被破解。不要受人为破解 Linux 和其他 UNIX 变种平台的口令速度很慢这一点的迷惑，对/etc/shadow 文件依然可以进行快速的字典攻击。

Nutcracker 或许是速度最快的 UNIX/Linux 口令破解器，这个工具由 Ryan Rhea 开发。由于 Nutcracker 是一个字典破解程序，因此要求使用字典文件。Nutcracker 本身携带了一个包含 2400 个词汇的示例字典，但必要时应该创建自己的字典。

### 3. Snadboy Revelation

很多应用程序都提供了保存口令的选项，这个做法很危险，原因在于能登录计算机的任何人都可以不加认证地登录到这样的应用程序中。更大的危险在于，人们经常重用口令，这就是说，如果有人掌握了一个人在某个应用程序中使用的口令，那么它就很有可能使用这个口令登录到此人所使用的其他应用程序中。

Revelation 是一个提取隐藏口令的工具，隐藏的口令通常在界面上显示为一连串的星号(*)或 X 号。假如一个应用程序使用了用户名 andrew 和口令 1r66nbg。虽然它的口令被显示为一串 X，但 Revelation 依然得到并显示了这个应用程序的口令。由于人们通常会重用口令，因此可将这个口令应用到该用户会使用的其他应用程序上，如电子邮件或记账软件上。

### 4. Boson GetPass—破解 Cisco 路由器口令

本节前面介绍的内容都是阐述 UNIX 和 Windows 系统中口令破解的工具。网络上的其他主机也有口令，特别是，Cisco 路由器包含了恶意黑客能够破解并获取访问权的口令。为了评估恶意攻击成功的可能性，渗透测试人员也必须对这类设备的口令进行破解。

在介绍如何破解 Cisco 设备口令之前，我们需要了解 Cisco 口令的工作方式。Cisco 路由器具有两种运行模式：

1) 用户执行模式(User exec mode)——用户执行模式就像旅店的大厅，可以进去看一看，但做不了什么事情。在用户执行模式下可以查看界面的状态、路由表以及执行其他信息收集任务，但不能进行配置；

2) 特权执行模式(Privileged exec mode)——特权执行模式就像拥有了旅馆中能够打开所有房间大门的万能钥匙一样，可以做自己想做的一切事情。在获取特权执行模式后，就可以完整访问路由器的配置。在用户执行模式转移到特权执行模式时可以设置口令。因此，保护好这个口令对保护路由器的安全至关重要。进入特权执行模式时有两种方式提供口令：明文口令支持和秘密口令支持。

## 5.6.4　检测口令破解

根据攻击发生的类型不同，检测口令破解也可能是一件困难的任务。当黑客进行标准的暴力破解或字典口令破解时，他通常要向目标发送成千上万的可能用户名和口令进行尝试。当某个用户名/口令组破解成功后，黑客就会开始下一步工作，获取系统的访问。黑客破解口令的另一种方法是获取系统的物理访问，然后盗窃口令文件和数据库。如果所有这些工作都过于困难而无法完成时，黑客就会转向实施最基础的社会工程攻击。下面介绍能够用于观察口令破解攻击的一些位置和方法。需要注意的是，物理访问系统的危险极具破坏性，当黑客复制了包含口令的文件(如 Windows SAM 文件)后，整个系统就可以说已经被黑客控制了。

### 1. 网络流量

在交换网络中，如果没有支持 SPAN 端口的合适设备，监控网络流量就是件困难的事情。在克服了这个障碍之后，可以使用网络嗅探器将所有流量记录到硬盘上，用于以后的分析。前面曾经提到，口令猜测会在短时间内向目标系统发送成千上万的测试用户名和口令。比如，可让 Brutus 使用字典口令猜测方法猜测任何 Telnet 服务器的口令，如 Windows Server 或 PIX Firewall。在这样的攻击中，Brutus 能在最短时间内发送尽可能多的用户名和口令，通过运用网络嗅探器能够发现这类大量出现的攻击企图。在正常的运行过程中，登录并不集中，而是零散的。因此，当某台机器短时间内发出大量登录尝试时，就可以认为发生了黑客攻击。但是，这种方法需要耗费大量时间，因此通常在使用其他方法发现了攻击企图后再采用这个方法进一步跟踪攻击。

### 2. 系统日志文件

检测登录失败的另一个更好的位置是目标计算机的系统安全日志文件。当打开记录失败登录请求后，日志文件提供了登录尝试的详细信息，包括时间、日期、用户名等。典型情况下会看到大量的登录企图。

### 3. 应对账户锁定

在运用口令猜测过程中，黑客都会遇到一个“账户锁定”问题。默认情况下，不管用户账户尝试登录了多少次，标准 Windows 计算机都不会锁定用户账户。这样默认设置的系统是黑客攻击梦寐以求的系统，它可以花费数日时间来攻击同一个账户，如 administrator，直到成功为止。

素质良好的系统管理员会手工配置登录设置，如每发现 5 次连续的失败登录，就锁定用户账户 30 分钟。当脚本小子遇到账户锁定时，他们通常就会转移到攻击用户列表中的下一个用户。他们一直使用这个账户，直到该账户被锁定为止。最后，目标系统的管理员将会听到合法用户抱怨不能登录系统。如果很多人都在抱怨不能登录时，这就是口令猜测攻击正在进行的明显信号。这种方法也是针对办公环境进行拒绝服务攻击的一种卑鄙手段。当锁定了系统的每个账户之后，工作就会被中断至少 30 分钟，任何用户都无法登录，除非其间管理员手工解锁账户。那么黑客高手会怎么做呢？黑客高手最多锁定账户一次，或者根本就不会锁定账户。假设锁定设置为 5，黑客高手会缓慢地攻击 3~4 次，然后等待 30 分钟的时间，再进行下一轮攻击。经过一周或一个月的时间，他最终会获取系统的访问权。因此，即使已经设置了账户锁定策略，也要经常观察一下日志文件，看是否存在问题。

#### 4. 物理访问

检测口令文件(比如 Windows SAM 文件或 Linux shadow 文件)的物理丢失十分困难。当这样的文件失窃后，黑客可以在自己家中耗费任意长短的时间来破解其中的口令。口令文件丢失的标识包括：破门进入办公室、笔记本或其他电脑丢失、备份磁带丢失、陌生的管理员账户活动等。某次事故后不久，目标系统管理员发现用户在不正常的时间登录系统，这也表示黑客对口令文件进行了破解。

#### 5. 垃圾搜寻和按键记录

垃圾搜寻和按键记录也可以被归类为物理访问。垃圾搜寻包括从垃圾箱中翻检旧硬盘、便签及可以用于搜寻用户名和口令的其他东西。如果系统管理员某一天上班后发现停车场或办公室的垃圾箱周围垃圾四溅，通常就可能是业余黑客在垃圾中搜索用户名和口令。

按键记录器是黑客安装的一种软件或者是某种智能设备，它安装在计算机和键盘之间，看起来像键盘适配器。利用这些设备，黑客能够捕获键盘上发出的每一个按键。现在有一些可以检测软件按键记录器的软件，但硬件按键记录器则难以检测。作为管理员，应该每天巡视服务器的连接，以便发现可能安装的物理按键记录器。与其他物理访问相似，按键记录器也不容易被发现。

#### 6. 社会工程

最难监测的方法之一实际上是社会工程。黑客可以通过训练有素的技巧，使用电话直接询问用户的用户名和口令，借口通常是系统维护，没有经过安全训练的用户会轻信这样的电话，给出自己的用户名和口令。这样的方式更难以检测，只有等用户将这样的情况报告安全小组的管理员之后，才会发现丢失了用户名和口令，而发现这个事情时，往往系统已经被清洗一遍了。另一种形式的社会工程技巧是背后偷窥。

### 5.6.5　避免或减轻口令破解风险

避免口令破解很困难，也不容易介绍这方面的所有要做的工作。总之，基本上不能够完全阻止口令破解，但能使破解很困难，从而让它成为一条黑客不能攻击系统的途径。下面介绍一些应该在自己的系统中实现的实践，实现的程度依赖于具体环境的要求。

### 1. 口令审计

即使是没有几个人的办公室，也应该定期行口令审计。安全专业人员应该建议用户使用长度较长的复杂口令，因为口令过短则十分容易被破解。比如，一个简单的 7 字符口令数小时就能够破解出来，而要破解一个 14 个字符长度的口令可能要耗费数月的时间。审计有助于管理员迅速找到口令中的弱点。

### 2. 记录账户登录日志与账户锁定策略

基本上所有系统(包括 Windows 和 UNIX 操作系统、路由器、防火墙及其他可管理设备等)都实现了记录账户登录日志的功能。如果我们不记录失败登录尝试，那么可能永远也不会发现黑客从早到晚尝试进入系统的企图。路由器和防火墙的日志配置比较简单，参考一下相应的帮助文件即可。Windows 系统可以配置为在安全事件日志中同时记录成功登录和失败登录事件。总之，不管使用什么方法，一定要确保最终收集了所有服务器上的日志。

防止口令破解的一种不错方法就是用户账户锁定策略。例如，在账户锁定之前，李明每分钟能够进行 1000 次口令尝试，但在使用了账户锁定后，要尝试 1000 次口令猜测，需要花费 125 个小时。然而，黑客可以利用账户锁定的这个特性来发起 DoS 攻击，然后通过其他渠道获取口令，如通过运用社会工程。

### 3. 口令设置与物理保护

与账户锁定一起设置的选项还包括口令长度、历史及口令过期时间。口令越长、越复杂，口令被猜中的可能性就越小。微软 Windows 系统提供了这些选项设置，不过，在默认安装时，它们都是关闭的，也就是说，这些系统允许 0 长度口令，也不提醒用户更换口令。

怎么强调服务器和客户端计算机的物理保护都不过分。黑客所使用的所有工具都需要得到 SAM 数据库、shadow 文件的备份，或者从这些文件中提取出数据。在获得这些信息后，就可以进行脱机暴力破解了。当拥有了管理员权限后，可使用 pwdump3、pwdump6 这样的工具跨越网络或从本地主机上提取出 SAM 的内容。但是，如果黑客没有管理员权限，就无法从网络口令文件中得到用户名和口令。

### 4. 员工安全教育和策略

向员工提供口令破解的风险信息及口令破解的难易程度，雇员就有机会认识到口令的重要性。实现口令设置建议及更换口令的过程有助于击退破解者的攻击，甚至击败社会工程的攻击。社会工程攻击难以防御，特别是用户没有得到系统的风险培训时更是如此。不管公司的大小，都应该教育员工使用唯一口令、不把口令写下来、不向任何人披露口令及用户名。

在正常办公环境中，绝大多数员工在日常工作中都不关心安全问题，当然系统管理员例外。应该教育网络用户关注安全问题，并使每个元素都保持安全状态。办公室都应该实现安全警示制度。保护口令安全、减少口令破解机会的措施包括下面内容：

- 解释原因：负责安全的领导十分关注口令安全；屏幕保护口令很重要；应该使用比较长的口令；使用 BIOS 口令。
- 应该告诉用户避免发生下述事件：所有地方都使用相同的口令；使用任何形式的增量口令，如第一次使用了口令 abcd123，下一次使用 abcd124 等；把口令写到纸上；

在家里的计算机上和办公室的计算机上使用相同的口令；安装非授权软件；向其他任何人披露口令；对电话询问用户账户和口令问题作出回答。

员工安全培训有助于减少可能的社会工程攻击，帮助用户理解为什么要制定这样的规则。当用户理解了这些规则的内涵后，他们就更可能遵守这些规则。

# 5.7　会话劫持

在海盗电影中，我们经常看到，毫无准备的货船被海盗征服。这种劫持行为发生在货船进入海盗经常出没的海域中。会话劫持与海盗劫持货船的过程有几分像。目标根本不知道该会话已经被劫持，因此授予人们各种权限，就像人们授权主机一样。

会话劫持(Session Hijack)是一种结合了嗅探及欺骗技术在内的攻击手段。广义上说，会话劫持就是在一次正常的通信过程中，黑客作为第三方参与到其中，或在数据流(例如基于TCP 的会话)里注入额外信息，或者是将双方的通信模式暗中改变，即从直接联系变成由黑客联系。会话劫持是一种内网攻击手段，黑客可以通过破坏已建立的数据流而实现劫持。

本节介绍会话劫持的技术和工具，同时说明检测和抵御这种攻击的方法。

## 5.7.1　什么是会话劫持

现在打一个比喻：假设你去市场买菜，在交完钱后你要求先去干一些别的事情，稍后再来拿菜；如果这个时候某个陌生人要求把菜拿走，卖菜的人会把菜给陌生人吗？当然，这只是一个比喻，但这恰恰就是会话劫持的寓意。所谓会话，就是两台主机之间的一次通信。例如，使用 Telnet 登录到某台主机，这就是一次 Telnet 会话；浏览某个网站时，这就是一次 HTTP 会话。会话劫持就是结合了嗅探及欺骗技术在内的攻击手段。例如，在一次正常的会话过程当中，攻击者作为第三方参与到其中，他可在正常数据包中插入恶意数据，也可在双方的会话中进行监听，甚至可代替某一方主机接管会话。

会话劫持是一种接管两台主机之间活动会话的行为。这与 IP 欺骗不同，在 IP 欺骗中，我们假冒另一台主机的 IP 地址或 MAC 地址。使用 IP 欺骗时，依然需要得到目标认证。而在会话劫持中，接管了已经得到目标认证的会话。这里也可以假冒主机的 IP 地址或 MAC 地址，但它包括了比假冒更多的内容。由于被劫持主机已经得到了目标的认证，因此会话劫持对于恶意攻击者具有很大的吸引力。这样，恶意攻击者就不需要花费大量时间进行口令破解，他也不关心认证过程有多么安全。原因在于对大多数系统来说，完成认证后，就开始使用明文进行通信了。这种工作方式使得大多数计算机都会被这种类型的攻击侵入。

会话劫持攻击有两种类型：

1) 主动型：找到活动会话，并接管这个会话，从而攻入目标主机。

2) 被动型：这种类型的劫持攻击首先劫持会话，然后记录目标与主机之间的所有流量。主动型劫持总是首先完成被动型劫持攻击者。

另外，还要区分会话重放(Session Replay)和会话劫持。这两种攻击都是中间人攻击(Man-in-the-Middle)，但在会话重放中，中间人捕获数据包，并在发送给目标之前修改其中的数据。而在真实的会话劫持中，通过假冒源主机(或目标主机)、将 TCP 序列号修改为与源主

机和目标主机相匹配的序列号，从而接管 IP 会话。常见的情况是，在假冒源主机的过程中，对源主机发起拒绝服务攻击，使其离线。

经常使用的会话劫持工具有：Huggernaut，Hunt，TTY Watcher 和 T-Sight。

### 5.7.2　ACK 洪流问题

执行了几次会话劫持后，就会发现 ACK 洪流(Acknowledgement Storm)的危险，ACK 洪流很快就会充斥网络，从而潜在地摧毁网络。前面已经介绍了 TCP 序列号预测的重要性，当发送了一个错误的序列号时，接收方认为最后一个应答被丢失，因此就重发最后一个应答，作为响应，原始主机返回自己的应答，以便重新同步序列号。在正常 TCP 操作中，这种设计方案很理想，它支持了可靠的数据传输。但是，当恶意黑客或渗透测试人员注入了带有错误序列号的数据包时，在主机和目标之前发送的响应会以指数方式增长，从而造成网络阻塞。由于现在正在假冒主机的 IP 地址，因此 ACK 数据包被发送到原始主机，以便重新同步序列号。

### 5.7.3　检测会话劫持

会话劫持能做得难以检测，除非攻击者引起了严重的损害或故意要引起人们对入侵行为的注意，否则绝大多数情况下可以毫不引人注意。

在会话劫持过程中用户会发现一些现象，其中比较常见的如下：

1) 客户端应用程序的会话停止响应或看起来被冻结了；

2) 短时间内突发的网络活动，这些活动减慢了计算机的运行速度；

3) 客户端应用程序挂起一段时间，原因在于用户实际上在与黑客竞争会话资源，黑客也在向服务器发送数据。这种做法迷惑了应用程序，使其等待第 4 层的响应；

4) 网络变得很忙，原因在于，当被劫持的客户端试图向服务器发送更多不同的数据时，在原始客户端与服务器间形成了 ACK 洪流。普通用户甚至高级别用户都很少会报告后两种现象，因为这些问题看起来与其他常见问题很相像，如应用程序崩溃、服务器忙碌、网络负载很重等。

看到“挂起”应用程序的用户通常关闭该应用，然后再重新打开。与此同时，黑客或许已经使用真实用户先前创建的已认证会话收获了丰硕成果。安全专业人员能够使用几种工具来协助会话劫持检测，下面两节将要讨论包嗅探和入侵检测系统。与此同时，安全专业人员也经常浏览 SANS 或其他优秀的安全网站，寻找是否出现了可用的新工具。

需要说明的是，交换并不能彻底阻断会话劫持。但在交换网络中会话劫持的实现难度将大幅提高。会话劫持可以发生在任何操作系统上，它本身并不是一个操作系统的问题，而是一个 TCP 协议设计和实现的自身问题，其中 TCP 协议的主要目标就是确保可靠的数据传输。

#### 1. 使用包嗅探器检测会话劫持

要监视会话劫持可以使用包嗅探器，但如果不知道哪些流量对于监视会话劫持有意义，那么监视就会存在困难。需要监视三种类型的数据包：ARP 更新(重复)，使用不同 MAC 地址在客户端与服务器之间传输的帧以及 ACK 洪流。在检测会话劫持的过程中，要牢记这三种类型。

2. 使用 Cisco IDS 检测会话劫持

正如前一节所介绍的，检测会话劫持可以使用包嗅探的方法。但这种方法要求网络管理员实时监视网络流量，需要耗费大量的时间和精力，因此它并不适用于企业的实际应用环境。诸如 Cisco 基于网络的 4200 系统和 IDS 系统等入侵检测系统(IDS)都内置了一些能够检测某些类型会话劫持的识别标志。利用这些系统，网络管理员不必实时监视网络流量，节省了大量时间和精力。

### 5.7.4　阻止会话劫持

会话劫持是一件难以对付的事情，IDS 监控仅能基于流量模式的假定得到一种猜测。Cisco IDS 在监控 T-Sight 会话劫持方面做得不错，但在其他一些情况下，存在误报及漏报的现象。例如，如果原始客户端在会话劫持期间从不通信，抑或客户端连接在 ACK 洪流产生之前被重置，那么永远也不会触发 3250 攻击特征，从而攻击在后台发生。这种现象并不是 IDS 本身的缺陷，而是由于没有发送足够的可用流量来提供可靠的检测。阻止劫持的发生才是真正的保护。不管怎么说，仅仅靠观察数据包流量的方法实在太不可靠。网络管理人员可以通过加密或签名协议的方法来对付会话劫持，虽然这样做依旧不能从根本上阻止会话劫持，但可以有效增加会话劫持成功的难度。

## 5.8　木马和后门的运用

病毒、木马和后门应用程序是每个工作在信息技术领域的人的噩梦。每年病毒都造成了商业领域巨大的损失。Chernoby1 病毒、情书(I Love You)病毒、美丽莎(Melissa)病毒及其他病毒都造成了严重的经济损失。本节将着重介绍木马、病毒及后门应用程序。

### 5.8.1　木马和后门程序概念

木马：木马是特洛伊木马的简称，英文为 Trojan Horse，这个名字来源于古希腊传说，它是指通过一段特定的程序(木马程序)来控制一台计算机。根据 Web 百科联机字典的定义，木马是一种伪装为善良应用的破坏性程序，最阴险的木马类型之一是声称能够帮助你的计算机摆脱病毒却在你的计算机中安装病毒的恶意程序。木马通常有两个可执行程序：一个是客户端，即控制端；另一个是服务端，即被控制端。木马的设计者为了防止木马被发现，而采用多种手段隐藏木马。木马的服务一旦运行并被控制端连接，其控制端将享有服务器端的大部分操作权限，如给计算机增加口令，浏览、移动、复制、删除文件，修改注册表，更改计算机配置等。

按照木马系统的方式及对系统所造成的损害，可以对木马进行分类。目前主要有 7 种类型的木马：

- 远程访问木马：它为恶意黑客提供了进入被控主机的远程 Shell；
- 数据发送木马：它向黑客发送敏感数据，如口令、信用卡信息、日志文件、邮件地址、即时消息联系人列表等，这个木马可以寻找特定的预定义数据，或者安装键盘

记录程序，并将按键记录发送给攻击者；

- 破坏性木马：它设计用于销毁和删除文件，通常看起来更像病毒，但不被反病毒程序检测到；
- 代理木马：它将受害的计算机用作代理服务器，这为攻击者利用受害的计算机做他想做的任何事情提供了机会，包括进行信用卡欺诈及其他非法行为，或者利用该系统对其他网络和系统发起攻击；
- FTP 木马：它设计用于打开端口 21(FTP 传输端口)，并让攻击者使用文件传输协议(FTP，File Transfer Protocol)连接到受害的计算机上；
- 安全软件关闭木马：它用于在用户不知情的情况下终止或关闭安全程序，如反病毒程序或防火墙，这种类型的木马通常与其他类型的木马结合使用；
- 拒绝服务攻击木马：它通过发送无用流量来阻塞网络或主机，发起 DoS 攻击。很多 DoS 攻击(比如 Ping of Death 和 Teardrop 攻击)都利用了 TCP/IP 协议的限制。

尽管每个木马的目的都各不相同，但其运行方式都是相似的，即将自己隐藏在主机上，并执行侵入者指定的动作。

后门(backdoor)：也称作 Trapdoor，它以一种未列入文档的方式获取对程序、联机服务甚至整个计算机系统的访问。后门通常由编写当前应用程序的程序员编写，通常也只有这个程序员知道这个后门，因此是一种潜在的安全风险。

### 5.8.2　木马与后门程序的检测及预防

木马和后门程序的检测很大程度上依赖于它们出现时间的长短及其欺骗手段的应用。比较古老的木马，绝大多数都能够被各种反病毒工具或防火墙工具依据其特征检测到，新的木马和后门程序可能会持续很长时间不被检测到。检测恶意软件的工具有很多，常用工具包括：MD5 检查，监控本地端口，监控远程端口，反病毒和反木马扫描器，入侵检测系统。

预防木马和后门攻击终究归结为持续地监视系统。下面是一些应该采取的预防措施：

- 安装补丁：及时安装系统及应用软件的补丁可以保持这些软件处于最新状态，同时也修复了最新发现的漏洞。通过漏洞修复，最大限度降低了漏洞利用型木马在系统上植入的可能性。
- 安装入侵检测系统：入侵检测系统(IDS)并不能够真正地防止木马或后门程序，但这些系统有助于检测这类恶意软件，并检测在系统中防止后门的早起信号。检测到 ICMP 和端口扫描是即将受到攻击的第一个信号，管理员应该警惕未来可能受到的攻击。
- 安装反病毒和反木马扫描器：此类软件包本身并不能防止攻击，而只能检测已经安装在系统中的恶意软件。但是，利用这些软件提供的功能，有助于防止未来可能发生的攻击，或者删除系统中已经存在的感染程序。
- 安装防火墙：安装防火墙能够大大增加黑客攻击成功的难度。仅开放绝对需要的端口，关闭其他所有端口，这样将自己暴露在互联网上的横截面减少到最低限度。但是，当木马成功地获取防火墙保护的主机或系统的访问之后，防火墙就形同虚设了。
- 安装基于主机的入侵检测系统：这类软件包监视所有应用程序的活动和进出计算机

的每一个连接。基于主机的入侵检测系统通常允许核准或拒绝程序的执行及控制哪些应用程序能够访问互联网。

- 培养风险意识：通过对自己及员工的风险培训，能够大大降低成为牺牲品的风险。告诉每一个人避免安装可疑软件或从互联网上下载软件；仔细观察任何不寻常的程序或进程(要查看进程，可使用 Windows 任务管理器，也可以使用其他进程查看软件)。

# 5.9　常见服务器的渗透

上一节介绍了在渗透测试过程中能够用于目标主机的木马和其他后门程序，本节将介绍测试服务器漏洞的其他方法，主要介绍最流行的两个操作系统平台系列上的漏洞利用，它们是 UNIX 平台(包括 Linux)和 Microsoft 的 Windows 平台。

无论测试哪个服务器平台，一般来说，首先要使用漏洞扫描器。漏洞扫描器扫描主机，并与漏洞特征数据库进行比对检查。现在已知漏洞有数千个，因此期待渗透测试人员跟踪所有漏洞是件不现实的任务。漏洞扫描器通过扫描目标主机并与一些漏洞特征比较，协助进行漏洞测试，可将漏洞扫描器看成一种自动化的渗透工具。这些漏洞数据库从一些漏洞网站上定期更新，漏洞扫描器的好坏取决于漏洞数据库的好坏，因此必须定期更新漏洞数据库。此外，漏洞扫描器的扫描结果仅反映测试时系统的漏洞情况。如果系统中存在某个漏洞，而这个漏洞又没有登记在漏洞数据库中，扫描器就不会检测到这个漏洞的存在。渗透测试对于系统安全很有帮助，但也仅反映进行测试时系统的状况及人们对各种风险的认识程度。

另一个重要因素是漏洞扫描器的成本及它的攻击性。有些扫描器在 General Public License(GPL，通用公共许可证)约束下免费使用，另一些扫描器则在每次使用时需要支付数千美金(基于 IP 地址的软件授权)。攻击性水平直接反映了扫描过程中对生产主机造成的冲击的程度。某些扫描器能够对目标发起拒绝服务攻击，如果攻击成功，这样的攻击对生产网络环境将造成严重损害，因此，只能在得到书面授权时才可以采取这样的行动。无论什么时候，首先要在一个封闭的实验室环境中测试漏洞扫描器的攻击性水平，如果没有得到拒绝服务攻击的授权，人们当然不希望发起能导致拒绝服务后果的攻击。

## 5.9.1　UNIX 权限和根访问

互联网上运行的一些大型服务器都使用了 UNIX 系统。UNIX 有两种类型的用户账户：普通用户和超级用户。用户又被进一步划分为组，在分配权限时提供了额外的灵活性。在 UNIX 架构中，所有东西都是文件，比如目录、设备等，并将文件分配三种类型的权限：读(Read)，写(Write)和执行(Execute)。权限划分为三个部分进行分配：分配给超级用户或 root 用户的权限，分配给用户组的权限以及分配给普通用户的权限。

作为一名渗透测试者，目标是获取 root 访问权限，原因在于 root 用户拥有系统的最高权限，几乎可以做它想做的一切事情。为获取 root 访问权限，需要运用权限提升技术将普通用户提升为 root 用户。常用的权限提升技术主要包括：栈溢出漏洞利用，rpc.statd 漏洞利用和 irix-login.c。

在得到系统授权之后就要隐藏文件，以防被其他人检测到。实现这个任务的常用方法是

使用 rootkit，其中在 UNIX 和 Linux 系统中最常用的两个 rootkit 为 Linux Rootkit IV 和 Beastkit 7.0。

### 5.9.2 Windows 安全模型和漏洞利用

Microsoft Windows 使用了两种安全模型：域模型和活动目录域模型。NT4.0 及以前的 Windows 版本中使用了域模型。它由一个主域控制器和一个或多个备份域控制器组成。在主域控制器失效时，可将备份域控制器提升为主域控制器。主域控制器在其安全账户管理器(SAM，Security Accounts Manager)数据库中存储和维护了整个域的所有账户。一个企业中可以有多个域，通过信任关系建立它们之间的联系。活动目录(AD)模型在 Windows 2000 和 2003 中使用。这个模型是一种分层结构，网络资源放置在一个 Jet 数据库中，目的是便于维护。在活动目录模型中，可以建立多个域控制器，所有用户账户在域控制器之间复制。通过使用森林、树和组织单元来实现分层，组织单元中保存对象，如打印机和用户账户等。

无论采用哪种模型，底层内核在所有 Windows 服务器平台上都是类似的。然而，随着每一个 Windows 新版本的发行，微软都强化了它的服务器的安全性，使得恶意黑客对系统的攻击更困难。下面介绍 Windows 系统攻击的两种技术。

与 Linux 类似，如果没有系统管理员的访问权限，那么很多针对 Microsoft Windows 的攻击就不能发挥作用。要获得管理员访问权限，需要利用权限提升技术，将访问权限从普通用户提升到具备管理员权限的用户。权限提升的方法主要有两种：PipeUpAdmin 和 HK。

Windows 平台中最流行的 rootkit 是 NT Rootkit。NT Rootkit 由两个文件组成：deploy.exe 和_root_.sys，需要把这两个文件复制到目标系统上，并运行 deploy.exe，这样将安装新的服务_root_。与前面介绍的 Linux 下的 rootkit 不同，NT Rootkit 没有包括很多的工具或替换系统文件。取而代之的是，NT Rootkit 让测试人员选择要隐藏哪些文件，这些文件可以是木马，也可以是后门。

### 5.9.3 检测服务器攻击与预防服务器攻击

检测服务器攻击是一项无休止的任务。服务器或其他计算机可能受到各种形式的攻击，因此，实现一种单一的检测方法来完成所有检测任务是一个不切合实际的想法。比如，如果安装了防火墙来防止外部的网络攻击，但服务器对于内部网络攻击、病毒、应用程序缺陷以及服务器窃贼依然敞开大门。因此，网络管理员应该在所有可能造成影响的地方实施检测和预防方法。

从头开始预防服务器攻击是一件困难的工作，原因在于不仅操作系统需要安全保护，而且运行在操作系统之上的应用程序及服务器使用的网络也都需要进行安全保护。保护服务器的操作系统、应用程序及网络的安全是一个漫长的试验和排错过程，但是，制定一些合理的指导原则可以帮助网络管理员更轻松地完成这项工作。

## 5.10 理解和应用缓冲区溢出

首先，通过一个比喻来解释缓冲区溢出。假设有一个小公共汽车，它有一个司机，可以载 4 名乘客，司机负责控制小公共汽车的方向，负责沿途上下乘客。计算机中的缓冲区的工

作方式与小公共汽车相似，它有一个指针，其作用就像司机一样，控制着缓冲区末端进行操作。现在，假设一下子上来了 5 个乘客，但小公共汽车只能容纳下 4 个乘客和一个司机。如果这 5 个新的乘客上车取代现有乘客的话，那么乘客的 4 个座位及司机的一个座位都被新的乘客所占领。这样就有了一个新的司机，小公共汽车处于新司机的控制之下。这也正是缓冲区溢出利用所发生的事情：缓冲区被填充了更多信息，指针被替换为新指针，新指针指向了恶意黑客选定要执行的代码。

一般情况下，缓冲区溢出的主要原因在于程序缺少边界检查。

### 5.10.1　缓冲区溢出的概念

现代计算中使用了两种常用的缓冲区：栈和堆。这两种缓冲区都能够用于缓冲区溢出，但在栈上应用缓冲区溢出更常见。

栈是运行时动态分配、用于存储变量的一片连续的内存。栈随着数据的添加和删除而增长或收缩，它采用了后进先出的策略。向栈中添加数据时，称为压栈，将数据放置在栈顶；从栈中删除数据时，称为弹栈，从栈顶中删除数据。栈顶的位置使用扩展堆栈指针寄存器记录，在添加数据时，栈顶的地址变小。比如，在数据添加到栈中之前，栈顶地址为 0x8，那么向栈中压入一个 32 位值后，栈顶地址标为 0x4。因此，压栈时，处理器减少扩展堆栈指针寄存器的值，让栈顶移到位置更低的内存地址上。

有些应用程序需要能提供比栈空间更大的缓冲区，解决这个问题的途径是堆(Heap)。当需要较大的缓冲区或不知道包含在缓冲区中对象的大小时，就要使用堆。堆溢出的工作方式几乎与栈溢出的工作方式相同。与栈的不同之处在于，堆没有压栈和入栈操作，而是分配和回收内存。C 语言中使用 malloc()和 free()函数实现内存的动态分配和回收，C++语言使用 new()和 delete()函数来实现相同的功能。

一般来说，确定返回指针的准确内存地址对于程序员来说相当困难。为帮助发现这一地址，程序员就在自己的代码中放入了无操作指令(NOP)。NOP 指令让程序移动到下一行执行，而最常用的 NOP 指令是 0x90。通过把一串 NOP 指令连续放置，可扩展利用代码的长度，但它们并不执行任何具体操作。

### 5.10.2　防止缓冲区溢出

防止缓冲区溢出的责任由开发程序的程序员承担。这些程序员应该选用那些没有漏洞的编程语言，如 Java、Python 或 Perl 等。然而，许多程序、甚至操作系统都使用 C 语言编写，并且很多程序编写的 C 或 C++函数在操作数据的时候都没有进行边界检查。例如，使用 strcpy()可将超过目标能够容纳量的数据复制到目标中，这样就导致了缓冲区溢出。因此，防止缓冲区溢出的基本步骤之一是在自己编写的代码中不使用 strcpy()函数，而使用如 strncpy()这样的函数，因为这个函数限制了要复制数据的长度。

现在，人们已经开发了一些能够跟踪栈指针并防止栈溢出的工具。比如，贝尔实验室开发的 libsafe 就是不错的一个，它能将危险函数替换为 libc 库中安全版本的函数。更重要的是，Libsafe 的源码可在完全开放源码的 LGPL 许可证下获得。

一些函数能引起缓冲区溢出，但它们在一些程序中依然被采用。基于编译器的防止缓冲区溢出方案是在编译程序时添加一些辅助代码，通过更改程序实现防止缓冲区溢出攻击的目

的。在实现方法上有几种不同的选择：一种方法是对每一个返回指针加上一段边界检查代码，这种方法将造成可执行程序长度放大一倍的后果；另一种方法是程序StackGuard采取的方法：在返回地址的后面插入一个额外的字，称为密探，并在函数返回时检查这个密探字。如果它的值被修改，就发生了栈溢出，此时，StackGuard将这个事件记录到syslog服务器上，并终止程序运行。

让栈中的数据不可执行是防止缓冲区溢出攻击的另一种方法。比如，Solar Designer 为 Linux 发布了一个补丁，它能防止代码在栈中执行，这种方法极大地提高了防止栈溢出攻击的能力。但是，这种方法也存在一些缺陷，一些需要在栈中执行命令的合法程序将不能再正常工作。如果没有源代码，防止缓冲区溢出是件十分困难的工作，即使有了源代码，也不可能重写整个程序，当然更不可能完全避免缓冲区溢出问题。能够做的工作是，教育程序员编写更安全的程序，并以最快的速度安装现有应用程序的最新补丁。

## 5.11　拒绝服务攻击

如果曾遭遇过系统崩溃，那么就会深切地体会到数据丢失和系统不能工作的痛苦了，这也正是拒绝服务攻击(DoS，Denial-of-Service)的目标。拒绝服务攻击是恶意黑客通过某种手段使系统不能正常使用的一种攻击，当黑客不能够使用其他手段访问数据，抑或是仅仅为了出名、造成影响时，就有可能发起 DoS 攻击。

DoS 攻击可以划分为三种类型：带宽攻击，协议攻击，逻辑攻击。

带宽攻击是最古老、最常见的 DoS 攻击。在这种攻击中，恶意黑客使用数据流量填满网络，脆弱的系统或网络由于不能处理发送给它的大量流量而导致系统崩溃或响应速度减慢，从而阻止了合法用户的访问。简单的带宽攻击能利用服务器或网络设备吞吐量限制达到目的——发送大量的小数据包，快速发送大量数据包的攻击通常在流量达到可用带宽限制之前就淹没网络设备。由于路由器、防火墙、服务器都存在输入/输出处理、中断处理、CPU、内存资源等方面的约束，因此读取包头进行流量转发的设备在处理大速率数据包(每秒钟处理的数据包数)时面临压力，造成对数据量不敏感。现实中，拒绝服务攻击通常通过大速率数据包的实现，而不仅依靠大量的数据流量。

协议攻击是一种需要更多技巧的攻击，它正在变得流行起来。恶意黑客以目标系统从来没有想到的方式发送数据流，如攻击者发送大量的 SYN 数据包。第三种类型的攻击是逻辑攻击。这种攻击包含了对组网技术的深入理解，因此也是一种最高技术的攻击类型。逻辑攻击的一个典型示例是 LAND 攻击，这里攻击者发送具有相同源 IP 地址和目的 IP 地址的伪造数据包，很多系统不能处理这种引起混乱的行为，从而导致崩溃。

尽管发自单台主机的简单 DoS 攻击通常就能发挥作用，但如果有多台主机参与攻击效率就会更高，这种攻击方式称为分布式拒绝服务(DDoS，Distributed Denial of Service)攻击。很多防火墙和入侵检测系统在检测到活动的 DoS 攻击后会阻塞单台主机，但是如果有上万台主机参与攻击，想要阻塞它们几乎不可能，因为几乎没有防火墙能处理这样大的通信流通。虽然委托单位可能会要求渗透测试人员对某台主机做 DoS 攻击的测试，但极少会要求进行分布式 DoS 攻击测试。因此，本节将重点放在与渗透测试相关的 DoS 攻击。

### 5.11.1　检测 DoS 攻击

常见的 DoS 攻击有：Ping of Death 攻击，Smurf 和 Fraggle 攻击，LAND 攻击和 SYN Flood 攻击。这些攻击大多数都能应用到 DDoS 上。

DoS 攻击的检测通常比较直观，但有时，这些攻击也难以在一开始就发现。发生 DoS 攻击的典型症状包括：频繁的网络攻击，很高的 CPU 利用率，计算机无响应以及计算机在不确定的时间崩溃。DoS 攻击旨在阻止合法用户访问他们所需的服务，因此为了检测这种攻击，可使用防火墙和网络异常行为检测器等多种设备和工具。

### 5.11.2　防止 DoS 攻击

预防是阻止 DoS 攻击的关键。如果攻击截面大幅度减少，将显著降低受 DoS 攻击影响的可能。但是，阻止所有的攻击是不可能完成的任务，人们能够做的就是加强系统的安全性，并希望达到最好。为减少攻击漏洞，网络管理员应该进行检查的基本事项如下：安装服务包和修补包，只运行必要的服务，安装防火墙，安装入侵检测系统，安装反病毒软件以及关闭穿越路由器和防火墙的 ICMP。

通过安装服务包，能最大限度地减少应用程序和协议被攻击的机会。微软会定期发布修复安全漏洞的服务包和修补包。而强化系统的安全性则包括两部分，强化网络设备的安全性和强化应用程序的安全性。

## 5.12　软件漏洞

### 5.12.1　设计漏洞与实现漏洞

软件安全漏洞可分为两大类别：设计漏洞和实现漏洞。

如果程序设计不合理，那么程序将永远不可能满足其需求和目标。如果不遵循这里所述的经验，即便程序有着广受好评的安全设计，仍然包含实现方面的缺陷，而这些最终将导致软件容易受到攻击。软件设计要解决的问题是，为满足软件需求所需的任务，程序的各组件彼此间如何进行交互。它将规定系统构架，确定各组件可能的状态，并详细说明各组件之间的数据流。软件设计必须明确安全模型的结构，如果程序有要求安全机制的功能特性，那么必须在设计中规定安全机制如何运作。例如，在一个多用户程序中，其设计会规定用户如何进行身份鉴别、如何授权以及如何对其活动进行审计。几乎每个获取输入的程序都需要有安全设计来规定如何缓解威胁。设计漏洞属于设计错误，它使得编程人员即使编写完美代码也无法实现程序的安全运行。设计漏洞常存在于软件的安全功能特性中，例如，当使用密码技术时，却常在与安全功能并不直接联系的系统其他部分中出现密码相关的东西。安全的设计要求有经验并经过培训，而许多软件工程师却没有在安全工程技术方面得到适当的指导。

实现漏洞是由软件实际编码中的安全缺陷造成的。它们就像是复杂的喷气式飞机某部分中精密铸件上的微小裂纹或瑕疵。无论这架飞机设计得多么好，在某种环境条件下，施加特定的压力，这种微小的裂纹将会导致飞机失事。在软件中，一般情况下这些缺陷看起来就像

是马虎的编码行为。它们往往看上去微不足道，比如忘记了字符串变量是 Unicode 字符类型的(每个字符占用两个字节)，在计算缓冲区的时候本应是偏移量为 2，但在实际编码时却按照偏移量为 1 错误地计算缓冲区。然而，这些微小的错误如果恰好出现在不当位置又恰好被攻击者发现，那么它将变成一个十分危险的安全漏洞。优秀的安全软件设计人员会使用诸如分类法、最小权限、攻击面缩减、密码技术等安全技术，将实现缺陷的影响或严重程度控制到最低限度。分类法就是使用高度的抽象和健壮的接口验证，以确保模型的正确使用。最小权限就是只授予用户或进程完成其工作所需的最低权限。攻击面缩减是指只保留那些完成软件功能绝对需要的接口，非必需接口予以取缔。如果去除低优先级的功能有可能减少攻击面的话，则去除这些功能性。但是，对于商业软件来说，这样做有些困难，因为对于这些低优先级的部分是否最终成为软件的特性，市场的考虑要胜过工程的考虑。密码技术可用来对数据提供保护，这样即使某个安全机制失效，由于无法解密数据，攻击者也只能无功而返。优秀的安全性设计会交叉使用多种技术。

如果软件在设计时缺少对安全的关注，那么在后期就很难或者几乎不可能使之成为安全的软件。另一方面，安全的设计也不能防止或减缓所有的实现缺陷。所以，即使设计堪称一份杰作，实现细节也依然重要，必须坚持安全的编码标准，实现的东西必须经过测试。

## 5.12.2　常见的安全设计问题

### 1. 密码技术使用中的隐患

密码技术是用来创建安全系统的一种基本技术。当双方需要在像 Internet 这样的不安全通道上进行通信时，窃听就是一种威胁，而密码技术可用来应对这种威胁。当系统需要对用户进行身份鉴别时，密码技术可用来安全地存储口令或令牌；当要求在像 HTTP 这样的无状态协议之上维护会话状态时，密码技术能够安全地维护会话。然而，密码技术是一种充满隐患的复杂技术。接下来将介绍三种常见隐患。

#### 1) 密码技术选用不当

一般情况下，应该使用经 FIPS(Federal Information Processing Standards，美国联邦信息处理标准)认可的算法，因为这些算法经过了密码阻止的严格审查，而且包含了所需要的各种类型的算法。对称密码技术和公钥密码技术是两种主要的密码技术。在对称密码技术内有块密码和流密码，块密码要求选择一种操作模式。对称密码技术要求在发送端和接收端均使用固定的密钥，因此，对密钥的管理就成了一个问题。公钥密码技术允许发送端和接收端分别使用自己的密钥，在常见的 SSL(Secure Sockets Layer，安全套接层)中，就一起使用了这种技术和对称加密技术。密码散列函数和消息验证编码比较适用于侦察篡改行为或进行消息签名的安全机制。

#### 2) 依赖隐蔽式安全

对于程序员所以依赖的安全机制来说，隐蔽式安全是很有吸引力的。当且仅当没有人付出足够的努力来突破这种安全机制的时候，隐蔽式安全才能起到作用。即使这种机制实现得确实很好，也只能在某段时间内安全，这段时间究竟有多长，这是攻击者所控制的。如果攻击者花足够的资源，即使是实现得最好的隐蔽式安全也终将失效。

#### 3) 编写到程序中的密钥

密码技术的基本内在法则之一就是密钥必须一直作为秘密存在，否则整个密码系统就将失效。许多程序会有这样的需求：数据必须安全地存储在磁盘或数据库中；可能也有这样的需求：在服务器和客户端之间，应用程序之间传递的数据必须经加密后传输。密码系统不容易构建，除非具有多年密码技术的经历，否则要想编写自己的加密算法几乎是不可能的，即使使用现成的密码技术也并非易事。密码技术中最具挑战性的部分之一就是密钥的管理，也正因为如此，才将密钥编写到程序中。

密钥管理就是保存密钥并将密钥来回传递以使密码系统运作的过程。程序员往往遗漏这个关键因素，或者没有足够的时间来做好这件事情，所以他们将密钥当做一个静态值编写到应用程序中，并试图通过 XOR(异或)编码或其他二进制匹配方式将整个值进行干扰处理，但事实上于事无补。对于任何拥有源代码或者编译后的二进制文件的人来说，这样编写到程序中的密钥都是可用的。经过少量的反向工程，攻击者就可解密那些存储在应用程序中的数据或从网络中觉察到的加密数据。一条适用的经验法则是：如果把密钥编写到应用程序中，就需要将整个应用程序当作秘密来对待。但对面向大量用户分布部署的应用程序来说，这显然是不可行的，因为如果这样的话，用户就将知道其他用户的密钥，这正是人们说数字版权管理(DRM，Digital Rights Management)从根本上存在缺陷的原因。

#### 4) 错误地处理私密信息

私密信息是指除了向授权访问的用户开放外，应用程序不应该公开的那些数据。诸如登录密码、登录密码散列、密钥以及会话标识符之类的秘密信息是非公开数据中最关键的信息，因为这些数据的暴露通常会破坏系统的安全性。另外一些私密数据，比如账户名称、信用卡号以及账户的收支情况等，这些通常都是非常重要而且需要保持其私密性的数据。系统设计人员应该对系统的数据资产价值进行评审，这样除了上述信息之外的其他有价值的信息也就不会被公开了。

对私密信息的恰当处理就是不要把这些信息随意地写到临时文件、日志文件或者交换磁盘空间中。保存过私密数据的缓冲区应该擦除，这样如果系统为其他用户会话而重用内存，后来的用户就不会看到之前他人的私密信息。如果数据是写入一个文件或数据库中，那么应该对数据进行加密，这样即使系统安全性遭到破坏，攻击者也不会得到明文的私密信息。

### 2. 对用户及其许可权限进行跟踪

许多联网的应用程序都需要跟踪以确定哪个用户与某个事务相关联，以便在该事务完成之前，应用程序能够执行一个授权步骤。如果这个用户未获得该事务或该事务所操作数据的授权，那么应用程序必须犯一个错误。

#### 1) 会话管理薄弱或缺失

为用户创建会话后，必须对其进行管理。典型的情况是 Web 应用程序通过为会话分配一个不可猜测且不可预知的会话标识符并将其存储在 Cookie 中，从而进行会话的管理。在会话管理中，应使用密码散列，如果会话标识符使用简单的增量数字或者时间戳的话，攻击者就会猜出有效的会话标识符，并使用其他用户已经认证的会话。此外，会话标识符在网络上的传输必须使用某种加密手段，如 SSL。这是因为，如果攻击者能嗅探网络，或能有效实施

代理攻击的话，这个攻击者就可以访问到其他用户的会话标识符。最后，应用程序还必须不存在跨站点执行脚本漏洞，因为这种漏洞可用来窃取用户的 Cookie。

#### 2) 身份鉴别薄弱或缺失

由于授权依赖于身份鉴别，故身份鉴别本身必须是安全的。用户发送到系统的密码必须通过像 SSL 这样的安全连接来传送，这样密码就不会被中途截取。身份鉴别步骤不应存在被绕过的可能性，在系统开始慢下来之前，密码的质量必须强制满足不允许攻击者尝试密码的次数超过三次，这样就可以防止暴力攻击。

#### 3) 授权薄弱或缺失

对于大多数应用程序来说，正确地实现授权并非易事。应用程序中不能存在让攻击者绕过授权步骤并通过系统执行未获得授权的事务的可能性。

### 3. 有缺陷的输入验证

缺少输入验证或输入验证存在缺陷，是造成许多极严重漏洞的头号诱因。这些漏洞包括缓冲区溢出、SQL 注入以及跨站点执行脚本。输入验证非常关键，必须对其缺陷加以纠正，应该花时间对输入验证例程进行审查，并针对其正确性进行测试。

#### 1) 没有在安全的上下文环境中执行验证

许多客户端/服务器应用程序都在客户端执行输入验证，以提高性能。如果用户输入了错误数据，客户端验证就可以较快地对数据进行验证，而不需要将数据通过网络发送给服务器来完成验证工作，这是在客户端进行验证的好理由。但是，如果仅在客户端进行验证的话，那么攻击者通过禁用客户端验证所在的代码部分(如 JavaScript)，就可以很容易地绕过这个验证步骤。他使用一个自定义 Web 客户端或者使用 Web 代理服务器，就可以操纵客户端的数据而无须进行客户端验证。

#### 2) 验证例程不集中

验证例程不容易写，程序内位于各层次上的许多例程会带来复杂性，难以进行正确性审查。输入验证应该尽可能地在靠近用户输入的位置执行，并且要集中，这样就可以确保所有数据都会经过验证例程，并能确保输入验证本身的正确性。应该注意的是，有时简单地将相互关联的例程进行集中，会形成过于简化的验证例程，最终变得验证不充分。当通过一个验证器例程集中对外部数据流进行验证时，必须主要保持严格的验证。

#### 3) 不安全的组件边界

现在的许多软件都是用多个组件来构建，图表控件就是一个组件的例子，该控件作为一个数据库对象，用作和数据库服务器连接的接口。这些都是现成的对象，一般由组件或数据库厂商提供。开发人员往往会将代码拆分为多个组件，这样就可以在其他软件项目中重用，或者通过这种组件化的方法简化其程序内部的通信。例如，一个应用服务器可以有一个配置组件，管理员可以使用它通过独立于应用服务的另一个进程对应用服务器软件进行配置，这个组件可以通过注入命名管道等进程间通信方式来与应用服务器引擎进行通信。

组件边界是指程序的位置点，各组件在这些位置上进行通信。典型情况下，组件间的通信是通过 TCP/IP 套接字、命名管道、文件、共享内存或者通过远程过程调用来完成。如果不

对这种通信通道进行身份鉴别，所有的组件间交换的数据就是潜在的敌意数据，这是因为通信通道很可能存在注入敌意数据的可能。许多组件接口都假定组件间通信的数据经过了验证，联系这一实际情况，这就使得组件边界漏洞随时可能发生。因此，对数据的验证应该在每一个组件边界上都执行，这通常称为“防御性编程”。组件边界越多，意味着必须正确验证的输入点越多，同时也意味着程序员出错的几率也越高。

### 4. 薄弱的结构性安全

攻击面就是攻击者可能与之进行交互的应用程序边界。对于大多数应用程序来说，这会是应用程序所具有的网络接口，不过，攻击面也有可能是文件系统、注册表或进程通信。应用程序与外部通信的方式越多，其攻击面就越大。一个在两个端口上对网络进行监听的程序通常就不如在一个端口上监听的应用程序安全。

当然，攻击面的大小不仅是数量问题，还与攻击面如何实现有关。对于一个程序来说，如果在执行安全性功能(如身份鉴别、授权或输入验证)之前执行的数据输入处理工作越多，那么其攻击面就越大。在执行上述安全性功能之前，对数据进行大量的解析或进行计算，就意味着会有更多代码，就会存在更多攻击者可访问的漏洞的可能性。

当一个漏洞被利用时，攻击者无论是执行任意代码或者是操作该系统的文件系统，其动作都是以那个存在漏洞的运行进程所具有的权限来执行的。这就是攻击者力求找出那些运行于高权限级别的软件(UNIX 系统上以 root 级别运行，而 Windows 系统上以 administrator 或 SYSTEM 权限级别运行)的漏洞，并对其加以利用的原因。以尽可能小的权限级别运行来达到软件所需，这样即使出现漏洞并被利用，也可以限制其带来破坏的影响。

安全设计的基本内在法则之一就是不要依赖一种机制来实现安全性，因为那种机制可能会失效，应该并用多种相互独立的安全机制。银行的物理安全是实现生活中纵深防御的一个例子：金库不把门开在大街上，银行有一个带锁的前门，一个带锁的室内门，而后才是最后的金库大门。每一层门上都安装了安全摄像机和警报器，在两门之间，房间内可能还会有移动物体传感器。这些安全机制的任何一种都可能失效，但整个安全系统仍然可以用来降低银行遭抢劫的风险。

当为应用程序设计降低安全威胁的措施时，设计人员应该使用多种机制与安全威胁抗衡。对于 SQL 注入威胁，设计人员应该使用输入过滤来确保数据(可能最终会作为 SQL 查询的一部分)格式良好。此外，还应该使用存储的预先准备的语句，这些语句汇集了所有查询的变量。应该对敏感数据进行加密，这样即使攻击者获得了数据访问权，他也需要正确的密钥。事务处理应该在日志中记录 IP 地址，另外，如果可能的话，还应该记录与之关联的用户名。引入到软件中的每种威胁缓解措施都有其局限性，还有可能存在缺陷。但如果使用了纵深防御，软件即使面临着某项威胁缓解措施的失效，也仍将是安全的。除非攻击者窃得身份凭证，否则他很可能会企图让应用程序去做那些设计中原本不打算做的事情。这种非预期的功能主要来源就是错误处理代码，这些代码往往在设计和测试方面有所不足。作为一个安全测试人员，应该重视错误处理代码，因为程序的大部分输入都将试图使得程序不去执行其预定的功能。如果程序不能做到对任何输入都统统非常得体地应对，那么攻击者就会利用错误处理程序例程中的非预期功能来改变程序的执行。

另一个错误处理问题就是，如何报告错误情况而不会暴露为攻击活动提供指引的系统相

关信息。错误消息中往往包含堆栈跟踪信息、目录信息以及软件所连接的软件组件或服务的版本信息等。软件应将错误细节记录到管理员可以读取的日志文件中，并只是简单地告诉用户发生了一个错误。

#### 5. 其他设计缺陷

多层应用程序往往会在一个进程之间传送命令或其他解释性代码。如果攻击者能够找出修改这类代码的途径，那么他就会找到办法来修改应用程序的执行。Web 应用服务器是一个常见的例子，它会向 Web 客户端发送 JavaScript 代码来执行客户端呈现或者客户端验证，另一个例子是向数据库服务器发送 SQL 查询的应用服务器。

应用程序在设计时应该将用户提供的数据和程序自己执行或传给其他系统执行的代码进行隔离。代码是静态时，要实现上述目标十分简单，但这些代码往往是使用用户数据来生成的，如果是这种情况，就必须要小心地将用户数据加以净化处理(去除其中的有害成分)。这样，就可以避免攻击者修改由用户数据生成的代码而控制部分系统的执行。

应用程序中输入的每一份数据都可能是潜在的危险之源。软件设计人员往往清楚地知道需要对用户输入的数据保持警觉。然而，他们却经常忘记应该对应用程序的所有外部数据都保持警觉。这也许源于一种旧观念，那就是人的输入是会出错的，所以必须对其构建防护措施。而由其他进程或者系统输入的数据是由计算机生成的，所以格式良好，并且也是安全的。这种安全的错觉是假定不会有恶意参与者欺骗外部系统或者这些系统被攻陷的情况。当发生这种情况时，从外部系统输入到应用程序中的数据就和用户输入一样危险。因此，必须将所有外部输入都当做危险的，并对其做净化处理，以去除其中的有害数据。

即使使用经过净化的数据，也必须多加小心，不要在数据中赋予任何未经保证的信任。如果可能，要对外部系统进行认证。对于那些可匿名访问的外部系统(如域名服务器)，就不能依赖其数据来做任何安全方面的判定，即使这些数据对记录日志很有用。但不应该让攻击者借之对外部系统施加控制以使得应用程序做出其他安全判定。

在一个多层次系统中，应用程序跨越多个服务器而运行，每个服务器都应该对其他服务器进行认证。而且如果可行的话，最好使用加密措施来阻止假冒和代理攻击。在一次假冒攻击中，攻击者会用一个看起来相像的服务器来代替原始服务器，但是这个新的服务器却被攻击者控制。这种攻击可通过修改所使用的域名系统或者修改网络层来完成。代理攻击和假冒攻击类似，只不过代理攻击使用了一个代理服务器，利用它仅对影响攻击的数据进行修改。

应用程序应该总是以安全的操作模式作为其默认值，不应要求使用的时候更改配置来使之安全，因为这种更改经常被遗忘或者不能被正确地执行。攻击者就以这种不安全的默认配置为攻击目标，因为他们知道，许多系统都是以这种默认的方式来部署的。

最后，审计日志是安全的应用程序之基本要素。有了审计日志，就能够查明攻击行为。

### 5.12.3 编程语言的实现问题

每种编程语言都有其特质，如果在编写代码的过程中对此没有理解且没有进行相应的处理，就会存在缺陷。程序员必须避免使用某些编程语言的元素或其编程环境，以避免造成实现缺陷。而其他一些编程语言元素，也只有在理解其安全含义的用法时，才能被安全地使用。

### 1. 编译型语言：C/C++

在安全领域中，存在大量安全漏洞，而产生这些安全漏洞的原因可追溯到 C/C++的不安全使用。C 语言是作为“底层”和可移植的语言而设计得，这就使得操作系统易于用 C 语言来编写，而现今所有流行的 OS 都是用 C 语言来编写的。在这些操作系统上，其本地接口也是用 C 语言编写的，故许多在这些系统上运行的应用程序为了获得较高的执行性能，也都使用 C 和 C++语言来编写。由于 C 语言在历史上具有比任何编程语言都要多的安全相关问题，用 C 和 C++语言来编写应用程序也就带来了问题。此外，大部分构成 Internet“管件”的软件——域名服务器、邮件服务器、Web 服务器、路由器软件甚至于防火墙软件等，都是用 C 或 C++语言编写的，所以说 Internet 上有很多安全问题一点都不为过。遗留软件是人们仍使用 C 语言的另一个原因，这些软件只是一直在增加一些新的功能特性，并持续地重复利用。

在 C/C++代码中发现的一些常见安全问题主要有：

1) C 语言没有安全的本地字符串类型，也没有安全而易用的字符串处理函数；
2) 缓冲区超限会覆盖栈中的函数返回地址；
3) 预防缓冲区溢出；
4) printf 类型的格式化函数；
5) 整数溢出。

### 2. 解释型语言：Shell 脚本和 PHP

通过创建一个“增强”环境，使得许多操作可以在默认的情况下“幕后”进行，脚本语言(如用于 UNIX 的 shell 脚本或用于 Web 应用程序的 PHP)使得程序编写更加容易。这种方便性是通过将程序员与操作系统环境固有的细节相分离来实现的。通过这种隔离方法，节省了编程时间，也有助于使程序独立于操作系统。这种隐含的功能会对安全性产生影响，因而，要安全地使用脚本语言，就必须理解这种影响。

脚本语言都是“高级”语言。由于使用动态内存管理为变量分配内存空间，程序员不需要管理内存分配、回收以及数据复制(如字符串复制)的底层细节。其好处就是许多困扰 C/C++的缓冲区溢出问题都不存在了。最常见的缓冲区溢出源自 C/C++代码，其中涉及字符串处理的时候尤其如此。而像 Perl 和 PHP 这样的脚本语言，发生缓冲区溢出的风险相对较小，但这也并不是说使用脚本语言或者 Java 的时候就不需要检查输入长度。因为这类语言中仍然还有可能将字符串变量传递给其他应用程序，这就容易引发缓冲区溢出。

某些情况下，变量仅仅是“经过”一个程序。例如，可编写一段这样的 Perl 脚本，它接受一个窗体输入，比如一个用户名，然后只简单地将这个输入值交给另一个程序，而这个程序可能是用 C++编写的或者是操作系统自带的。上述的第二个程序会容易产生缓冲区溢出，所以攻击者仍可能搞破坏。只是这个问题不会影响 Perl 脚本，因为在这里它只是将输入中转传递了一下。因此，无论使用何种语言或何种函数，永远要对输入长度进行检查。

### 3. 虚拟机语言：Java 和 C#

编译为字节码并需要虚拟机来运行的语言，通常来讲比编译成本机代码的语言更安全一些。Java 和 C#就是被设计为编译成字节码的编程语言的典型，这种编程语言在设计的时候，就一直牢记 C 语言的风险史。Java 和 C#强化了输入安全，这意味着它们不能访问任意的内

存位置。如果一个 Java 程序试图超过数组边界来访问一个数组，就会抛出一个异常。而如果程序用 C 语言来做这个操作的话，就能对内存进行读、写操作。

### 5.12.4　平台的实现问题

平台就是程序在其中运行的环境。操作系统是其主要的组成部分，但平台还可以包括程序与之交互的一些常用的组件。每种操作系统都有其特质，对此必须理解，以免产生不必要的负面效应而导致出现安全漏洞。

程序依赖操作系统来与用户、网络以及文件系统进行输入和输出的交互，还要依赖操作系统来产生新的子进程以及与其他进程进行通信。从安全观点看，所有这些操作都是很有风险的。一个设计完善的操作系统会清除默认行为，提供文档来说明安全访问操作系统的 API 或服务的方式，并且指导程序员编写安全程序。遗憾的是，操作系统开发人员没有想透或者没有提供文档来说明许多现代操作系统的隐含的安全问题，比如符号连接、目录遍历、字符转换等。隐含的安全问题就留给了漏洞研究人员来发现。

### 5.12.5　常见的应用程序安全实现问题

在任何平台上使用任何语言编写程序，某些应用程序的安全问题都会出现。其中某些问题是由应用程序的某个组件接受的恶意数据引起的，这种数据在应用程序的另一个组件被当做了合法的代码。另一些安全问题是由于对涉密信息的不当处理造成的，这些涉密信息必须保持其作为秘密的特征，并且要用不可预测的密码系统来履行加密的使命。常见的应用程序安全实现问题有 SQL 注入和跨站点执行脚本。

### 5.12.6　开发过程中的问题及部署方面的薄弱性

在开发过程中遇到的问题主要有：

- 安全需求和前提条件的文档记录贫乏；
- 交流和文档的匮乏；
- 在开发过程中缺少安全过程。

部署是指拿到软件，并将其在产品系统上进行安全和配置的过程。这既可以由定制应用程序的企业内部 IT 员工来完成，也可以由使用打包的软件产品的最终消费者来完成。在这两种情况下，部署软件的人，都不属于开发团队，他们需要通过文档或者安全和配置程序的指导来安装软件。

开发人员在开发过程中会设想软件怎样去部署。他们往往会假定软件所使用的文件和注册表键值只会由这个软件修改，而文件和注册表的访问控制机制需要恰当地进行设置，以保护配置文件和注册表不会被篡改。当出现文件和注册表键值设置为完全可写，这就意味着该系统上的所有用户都可以对其进行更改。如果开发人员假定配置文件总是格式良好的，原因是它只能被这个软件本身修改，那么开发人员或许就会放弃有效性验证的代码。如果攻击者发现了这个许可权限的薄弱性，他就可以篡改这个文件，企图由这个软件引入错误的处理，而这个软件会假定这种修改仍然是格式完好的数据。将文件的许可权限设置为“前提条件在部署配置中都成立”，为确保这些前提条件都符合，建议用文档记录这些前提条件，然后进行核查。

部署方面的另一个主要问题就是将软件安装为运行时具有不必要权限的情况。许多开发人员在构建软件时都图方便，所以他们构建的软件需要以 UNIX 上的 root 用户或者 Windows 上的本地系统用户的身份来运行。这些账户都是系统中具有最高权限的用户，因此如果这个软件有漏洞，并且被发现利用的话，攻击者就能获得对整个系统的完全控制权，这是当今许多服务器软件产品中存在的主要问题。即使开发人员耗费大量的时间来构建软件，以确保软件可以最低权限的用户身份来执行，但很多部署脚本依然将软件的运行身份设置成了 root，或者安装这个软件的用户将其安装为以 root 身份运行，而没有考虑其中隐含的安全问题。

### 5.12.7　漏洞根源分类

以上对那些最为严重和最为常见的软件安全漏洞根源进行了概述，当然对此并未做到毫无遗漏的穷举。现在，美国政府的国家标准技术研究所(NIST)正和一些行业组织一起努力构建一套针对所有漏洞根源的分类法，他们从几个源头(既有商业机构也有政府机构)出发征求信息输入来构建一个漏洞根源列表。NIST 的软件质量保障度量和工具评估项目力图协调以下五种最流行的分类法：

- 7 种有害领域。
- LCASP：综合的轻量级应用程序安全过程。
- 软件安全的 19 宗罪。
- OWASP 最为严重的 Web 应用程序安全漏洞前 10 名。
- CWE：常见弱点样表。

## 5.13　本章小结

本章主要介绍了软件安全动态渗透测试的相关内容。其中，首先介绍了动态渗透测试启动之前建立测试计划的步骤；接着介绍了动态渗透测试过程中常用技术手段，主要包括主机侦查、Web 服务器攻击、数据库攻击口令破解、会话劫持以及木马和后门的运用。另外，本章也对常见的服务器渗透攻击、缓冲区溢出攻击、拒绝服务攻击和软件漏洞进行了简要阐述，给读者将来对软件安全动态渗透测试进行更深入的学习打下基础。软件安全渗透测试是保障软件安全的主要手段，由于它模拟现实环境中软件可能面临的各种攻击，所以可以极大地提高软件的安全性。

# 第6章　WebInspect概述

## 本章导读

HP WebInspect 是一款可配置的自动化 Web 应用安全和渗透测试工具。它旨在模拟实际的黑客入侵和攻击，支持全面透彻的 Web 应用和安全漏洞分析服务。本章将介绍 WebInspect 的基础知识和安装方法。

应掌握的知识要点：

- WebInspect 的主要特点；
- 爬行和审计的概念；
- WebInspect 报告；
- WebInspect 手动黑客控制；
- WebInspect 的扫描策略；
- Web 服务评估；
- WebInspect 工具；
- WebInspect 新特征；
- WebInspect 软件安装；
- WebInspect 主要功能。

## 6.1　WebInspect 的主要特点

### 6.1.1　WebInspect 介绍

目前，许多复杂的 Web 应用程序主要基于新兴的 Web 2.0 技术，HP WebInspect 对这些应用程序执行 Web 应用程序安全测试和评估。HP WebInspect 可提供快速扫描功能、广泛的安全评估及准确的 Web 应用程序安全扫描。虽然传统应用扫描程序在发现 Web 1.0 技术中的漏洞时表现得很好，但是它们在扫描更新的 Web 2.0 技术时通常显得不够智能。

WebInspect 可识别很多传统扫描程序检测不到的安全漏洞。利用创新的评估技术，例如同步扫描和审计 (SCA，Simultaneous Crawl and Audit)及扫描并发应用程序，用户可以快速而准确地自动执行 Web 应用程序安全测试和 Web 服务安全测试。

WebInspect 是准确和自动的 Web 应用程序和 Web 服务漏洞评估解决方案。使用 WebInspect，安全专业人士和规范审计人员可以便捷地分析 Web 应用程序和 Web 服务。WebInspect 是由全球领先的 Web 安全专家每天维护和更新的唯一产品。这些解决方案专门设计用来评估潜在的安全漏洞，并提供所有用户需要解决这些问题的信息。

WebInspect 可提供评估技术的最新发展，能适应任何企业环境的 Web 应用安全产品。当用户开始进行漏洞评估时，WebInspect 的“评估代理”(Assessment Agent)能对 Web 应用的所有区域进行分析，当这些代理完成评估后，研究结果被发布到核心的安全引擎去分析，然后 WebInspect 启动审计引擎评估所收集的信息并应用攻击算法来查找漏洞的位置，从而确定其严重程度。通过这个方法，WebInspect 可持续使用适当的评估资源来适应用户特定的应用环境。

## 6.1.2　WebInspect 的主要特征

以下是 WebInspect 功能的一个简要概述。

### 1. 爬行和审计

WebInspect 可以用爬行和审计这两种基本模式来确定检测对象的安全弱点。

爬行是通过 WebInspect 来识别目标网站结构的一个过程。从本质上讲，爬行会遍历整个网站，直到没有更多的 URL 可以访问。

审计是目标网站的漏洞评估。当爬行和审计组合成为一个新功能时，称之为扫描。

扫描是基于实时审计结果的细化，产生整个 Web 应用程序的攻击面的全面视图。智能引擎采用了基于逻辑的、结构化的方法来分析一个应用程序，然后基于应用程序的行为和环境进行自定义攻击。WebInspect 把这些先进的、具有突破性的评估技术与存储在漏洞数据库中已知的 Web 应用程序漏洞相结合来进行扫描。

### 2. 报告

使用 WebInspect 报告可获得有价值、有组织的应用程序信息。在 WebInspect 的报告中，可以自定义报告细节，决定什么级别的信息包含在哪一个报告中，使报告适应每一个特定的观众。也可以使用报告设计器创建自己的报告，保存各种格式的报告，还可以使用图形方式来表达汇总后的漏洞数据。

### 3. 手动黑客控制

使用 WebInspect，可看到网站上实际发生了什么事件，也可以模拟一个真正攻击的环境。使用 WebInspect 能够查看任何页面中包含漏洞的代码，然后修改服务器请求并立即提交。

使用网络代理工具，如 Web 代理，从客户端接收请求，从服务器接收响应，或者找到满足自己创建的搜索规则的文本，可以暂停客户机到服务器的数据流。

### 4. 总结和修正

扫描过程中，WebInspect 为所有的漏洞检测提供汇总和补救信息。其中包括参考材料、补丁链接、预防未来问题说明和漏洞的解决方案。随着新的攻击和非正常使用的发生，WebInspect 也将总结和修正信息数据库。WebInspect 工具栏使用智能更新和最新漏洞解决方案信息来更新数据库。

### 5. 扫描策略

WebInspect 可修改和自定义扫描策略，减少 WebInspect 完成全部扫描所耗费的时间，以满足用户组织的要求。

用户还可以扩展 WebInspect 产品的功能，以满足组织的特定需求。可配置 WebInspect 来适应任何 Web 应用程序环境，并使用自定义检查向导来创建自定义攻击。

### 6. 可排序和定制的视图

当进行扫描或查看一个扫描时，在 WebInspect 窗口左边的导航窗格中包括网站、序列、搜索和步模式(Step Mode)按钮，这些按钮决定了导航窗格中显示的内容(或“视图”)。

- 网站视图展现 WebInspect 检测到的所扫描网站的文件层次结构。
- 序列视图根据 WebInspect 自动评估或手动爬行(步模式)的结果顺序，显示服务器资源。
- 搜索视图能找到满足指定条件的会话。
- 步模式能用来手动浏览网站，起始点为从网站视图或序列视图中选择的会话。

### 7. 整个企业范围内的应用能力

从整体企业的角度看，综合的评估过程为 Web 状态提供了一个全面的概览，使用户能有选择地对网络上所有基于 Web 的应用进行单独的或计划的应用评估。

### 8. Web 服务评估

WebInspect 提供针对 Web 服务漏洞的全面评估，通过 WebInspect 可评估包含 Web services/SOAP(Simple Object Access Protocol，简单对象访问协议)对象的应用系统。

### 9. 导出向导

通过 WebInspect 可配置的 XML 导出工具，用户能以一种标准化的 XML 格式导出扫描过程中发现的所有信息，包括注释、隐藏域、JavaScript、Cookie、Web 窗体、URL、请求和会话。用户可指定要导出信息的类型。导出向导还包括一个“清理”功能，可以防止导出信息中包含敏感数据。

### 10. 工具

在 WebInspect 中集成了一系列诊断和渗透测试工具，包括：

- 审计输入编辑器
- Cookie Cruncher
- 编码器/解码器
- HTTP 编辑器
- 日志查看器
- 策略管理器
- 正则表达式编辑器
- 报表设计器
- Server 分析
- Server 事件探查器

- 智能更新
- SQL 注入
- SWF 扫描
- Web Brute(网络狂人)
- 网络发现
- Web 窗体编辑器
- 网络 Fuzzer
- 网络宏录制
- Web 代理

### 6.1.3　WebInspect 新特征

WebInspect是首屈一指的 Web 应用程序安全评估工具，其设计专门针对当今复杂的、基于新兴的 Web 2.0 技术 Web 应用程序。这种新架构提供更快的扫描能力，更广的评估范围，以及对任意 Web 应用程序提供更准确的扫描结果。

#### 1. 闪存的静态分析

WebInspect 可反编译 Shockwave Flash SWF 文件的最新版本，然后对产生的 ActionScript 3 代码执行静态分析来检测漏洞，如不安全的编程实践、不安全的应用程序部署、违反 Adobe “最佳实践”的行为和信息泄露。

#### 2. 新报告系统

WebInspect 新版本的和强大的报告系统有助于分析演示数据。用户通过新报告系统可执行以下操作：

- 创建灵活的、可扩展的报告，并且采用改进的、速度更快的新一代工作流程；
- 修改标准报告或使用新的报告设计器来设计自己的报告；
- 从外部数据源中提取信息；
- 新报告样式编辑器自定义字体、颜色和背景等；
- 生成专业的扫描报告；
- 用新的会话报告重点分析单个会话。

#### 3. 可选的深度优先爬行

深度优先爬行最适合基于顺序浏览的这一类网站(例如在访问页面 B 之前必须先访问页面 A)。该方法会根据第一个网页 A 的第一个链接到达此链接指向的网页 B 的第一个链接，只有在访问网页 B 的所有链接后，再返回网页 A 去访问第二个链接，依此类推。与此相对应的是，广度优先爬行(也是另一种可行的方式)首先会访问网页 A 的所有链接，再逐次访问网页中每个链接所对应的网页。

#### 4. Java MVC 框架支持

通过 HP DevInspectJava 团队的深入研究，WebInspect 已经能够使用基于路径攻击和导航参数的深度优先爬行方法，支持建立在 Java MVC 平台上的应用程序。

### 5. 使用 IBM Rational ClearQuest 集成

用户可以直接将漏洞缺陷发送到 IBM Rational ClearQuest 版本 7 中。

## 6.2　软件安装

### 6.2.1　最低配置

在安装 WebInspect 之前，请用户确保系统满足以下最低要求：

- 支持的操作系统：Windows XP 专业版 SP3(32 位)，Windows Server 2003 标准 SP2(32 位)，Windows Vista SP1 的(32 位或 64 位)，Windows 7 SP1(32 位或 64 位)；
- 处理器：单核 1.5 GHz(推荐双核 2.5 GHz 或更高频率)；
- 内存：1 GB(建议 4 GB)；
- 硬盘空闲空间：2 GB(建议 20 GB)；
- 网络：一个有效的 Internet 连接(推荐更新)；
- 显示器：1024×768(推荐 1280 x 1024)；
- 数据库：Microsoft SQL Server 2005 Express 版 SP3 或 SQL Server 2005 标准版 SP3(推荐 SQL Server 2008 R2 SP2)。

当使用 SQL Server Express 版，扫描数据不得超过 4 GB。对于较大扫描，或允许共享扫描数据，使用 SQL Server 2008 标准版。

- 平台：Microsoft .NET Framework 3.5 SP1(推荐 4.0)；
- 浏览器：Microsoft Internet Explorer 7.0(推荐 9.0)；
- PDF 支持：Adobe Reader 8.1.2 或 9.0(仅用于在导出的 PDF 中查看报告内容)。

不支持测试版本的操作系统，服务包以及第三方组件(比如 Microsoft SQL Server Expres s)。

当安装 SQL Server Express 版时，接受所有默认设置；WebInspect 默认实例名是 SQLEXPRESS。惠普还建议选择“隐藏高级安装选项”和“添加当前用户为数据库管理员”的复选框。

SQL Server 2008 的标准版可安装在本地主机或附近并列的主机上。在 WebInspect 中，可通过单击 Edit(编辑)→Application Settings(应用程序设置)→Database(数据库)来配置该选项。

### 6.2.2　下载和安装 WebInspect

#### 1. 下载

WebInspect 的官方网站是http://www8.hp.com/cn/zh/software-solutions/ software.html? software.html?compURI=1341991，登录该网站，单击试用版与演示的下拉菜单，即可下载目前最新版本的 WebInspect 10.20，如图 6-1 所示。

仔细阅读 HP WebInspect 10.20 软件的下载使用条款，单击“我同意”即可进入到下载界面，如图 6-2 所示。

可以下载时，可以使用 HP WebInspect Manager 或标准下载(即迅雷下载)，建议使用迅雷

下载，如图 6-3 所示。




图 6-1　WebInspect 10.20 下载界面




图 6-2　WebInspect 10.20 下载使用条款




图 6-3　WebInspect 10.20 下载界面

## 2. 安装

下载的 WebInspect 软件是 ISO 镜像文件，需要在 DAEMON Tools Lite 软件中打开，如图 6-4 所示。在联网条件下安装 WebInspect 时，如果本地计算机没有安装 Microsoft .NET Framework 4.0，系统会提示安装。在安装 WebInspect 时，系统检测计算机是否已安装 Microsoft SQL Server 2008 R2 Express SP2 或者更高级版本，有则直接安装 WebInspect；没有，系统先默认安装 Microsoft SQL Server 2008 R2 Express SP2，然后安装 WebInspect。

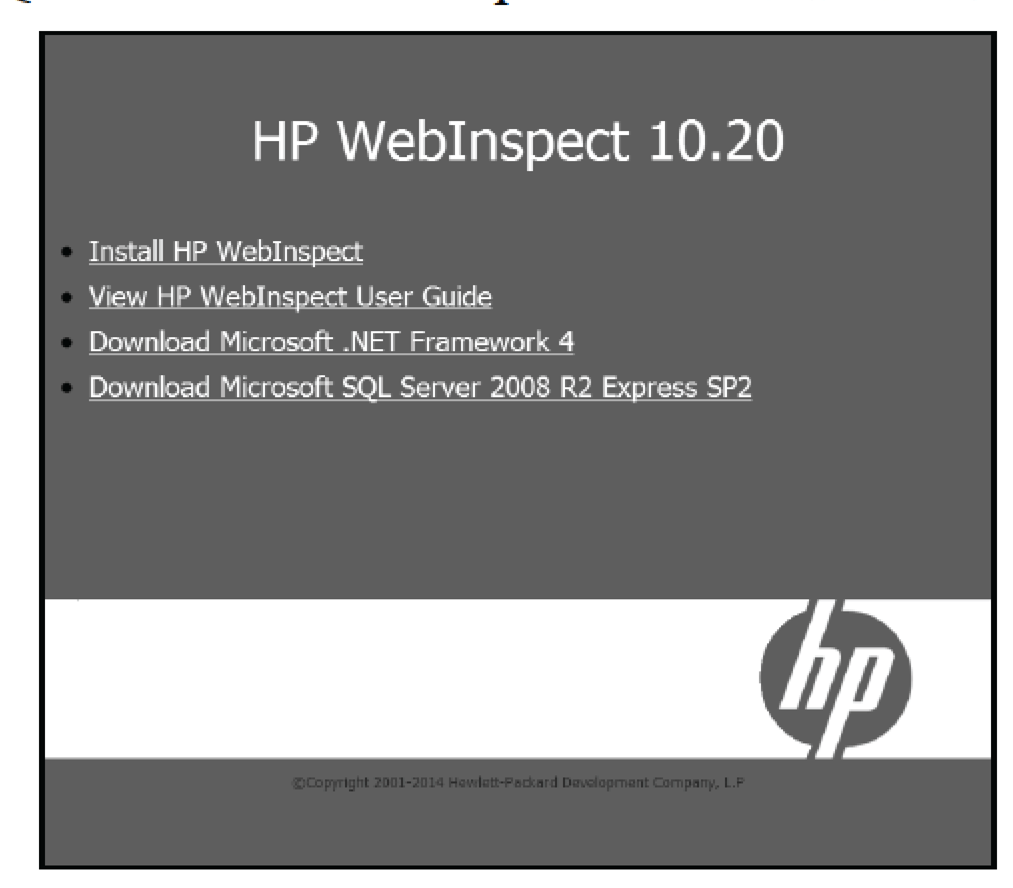



图 6-4　WebInspect 软件的 ISO 镜像文件在 DAEMON Tools Lite 软件中打开后界面

接下来按照下列步骤来安装 WebInspect：

1) 启动安装程序。

2) 在欢迎界面上，单击 Next(下一步)，如图 6-5 所示。




图 6-5　安装 WebInspect 10.20 欢迎界面

3) 查看许可协议。如果接受，请选中该复选框并单击 Next(下一步)；否则请单击 Cancel(取消)。

4) 在目标文件夹窗口中，选择要安装的软件文件夹，然后单击 Next(下一步)。

5) 如果要安装 WebInspect 作为评估管理平台(AMP)的检测部件，可参见图 6-6。

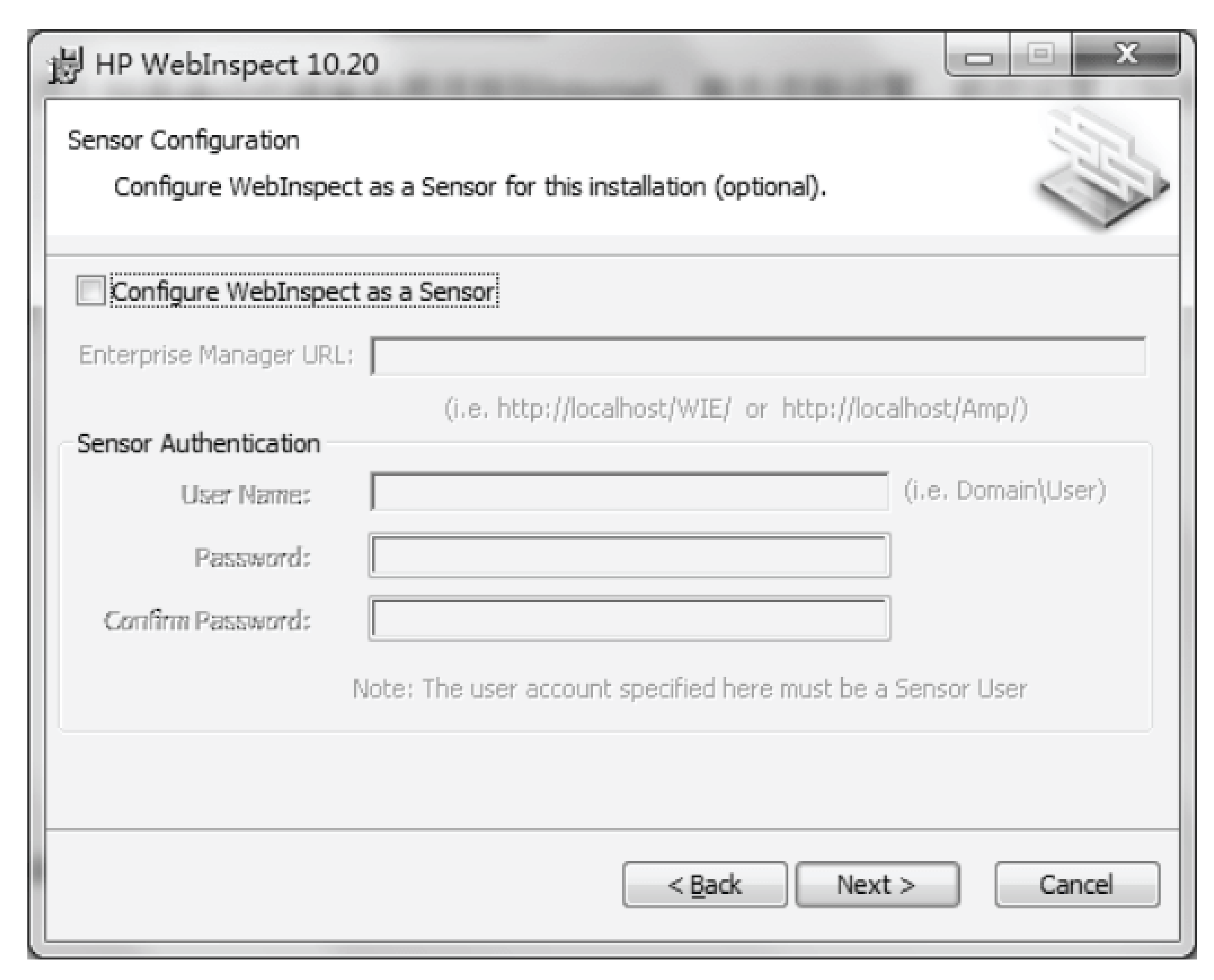



图 6-6　安装 WebInspect 10.20 配置界面

a) 选择配置 WebInspect 作为 AMP 的检测部件。

b) 输入 AMP 管理器的 URL。

c) 在检测部件验证组中，输入这个检测部件的 Windows 账户证书。

6) 单击 Next(下一步)。

7) 在准备安装窗口中，单击 Install(安装)。

8) 当这个过程完成后，单击 Finish(完成)，如图 6-7 所示。

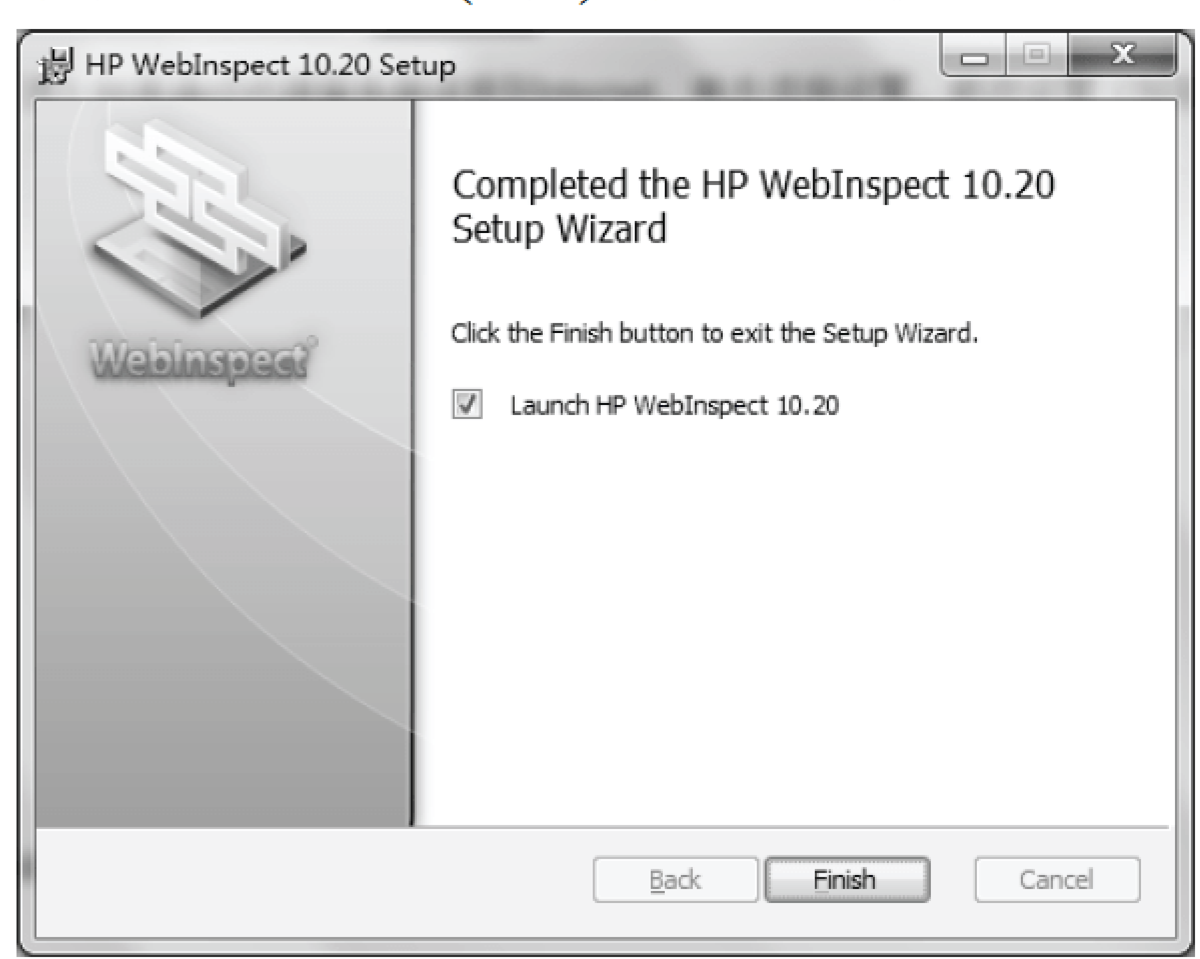



图 6-7　安装 WebInspect 10.20 完成界面

### 3. 许可证

首次启动 WebInspect，该程序显示 WebInspect 产品注册向导，它会提示选择下列选项之一，如图 6-8 所示。

- 现在激活；
- 注册一个 15 天试用版。




图 6-8　WebInspect 产品注册向导界面

### 4. 试用注册

请按以下步骤开始 WebInspect 免费的 15 天试用版。

1) 在 WebInspect 产品注册向导窗口中，选择 Register 15 Day Trial(注册一个免费的 15 天试用版)，然后单击 Next(下一步)。该向导将显示一个窗口，提示输入有关用户和用户的公司信息，如图 6-8 所示。

2) 输入所要求的信息。

3) 如果是通过 Network Proxy(代理服务器)连接到 Internet，从 Proxy Profile(代理配置)文件下拉列表中选择一个设置，修改设置(如有必要)，然后单击 OK(确定)。

4) 单击 Next(下一步)。

该步骤试图联系惠普服务器，惠普服务器将给用户发送一封电子邮件，电子邮件中包含 32 个字符的激活令牌。

5) 单击 Finish(完成)。

6) 当邮件到达时，单击 Edit(编辑)菜单并选择 Application Settings(应用程序设置)。

7) 在应用程序设置窗口中，从左边窗格中选择 License(许可证)。

8) 输入 32 位数的 license token(许可证令牌)，忽略任何可能出现在字符串的连字符(或简单地复制令牌，在令牌激活字段的第一个块定位光标，然后按 Ctrl + V 键)。

9) 输入一个描述(可选)。

10) 单击 OK(确定)。

### 5. 现在激活

如果用户从惠普的电子邮件中收到一个激活令牌，请选择此选项。

1) 在 WebInspect 产品注册向导页面，如图 6-8 所示，单击 Activate Now(立即激活)。该向导将显示配置 WebInspect 的许可证窗口。

2) 在许可证方法组中，选择下列选项之一：

- Connect directly to HP corporate license server(直接连接到惠普公司的许可证服务器)：许可证由惠普服务器控制。

- Connect to local HP License and Infrastructure Manager(连接到本地的惠普许可证和基础设施管理器)：许可证通过运行 HP 许可证和基础设施管理器(LIM)软件的本地服务器进行控制。此安装界面提供一个链接，下载 HP LIM。

3) 输入用户信息组所要求的信息，如图 6-9 所示(下面步骤使用此方式在已连接到 Internet 条件下激活)。

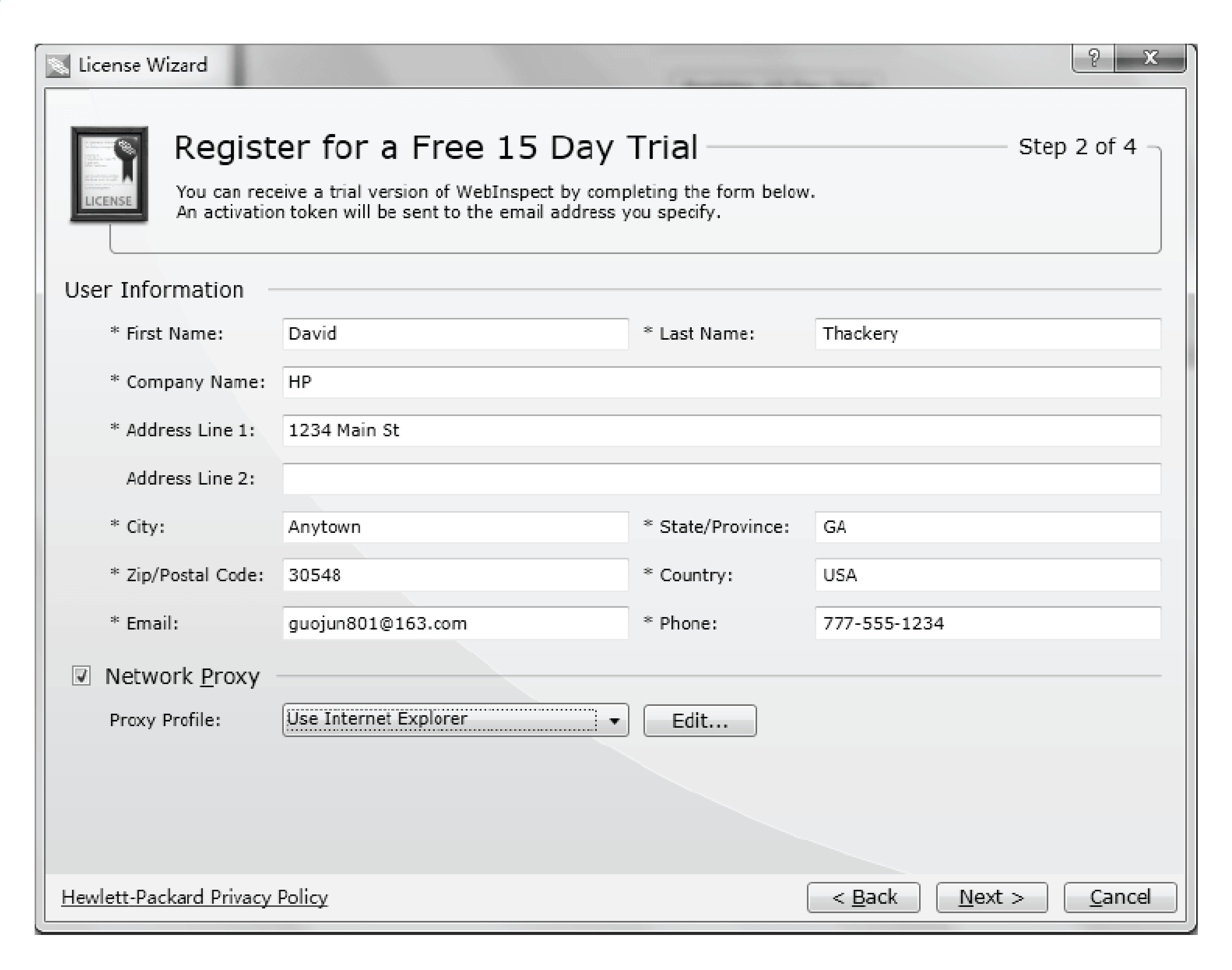



图 6-9　输入相关信息

4) 单击 Next(下一步)。

5) 输入 32 位数的 Activation Token(激活令牌)，忽略任何可能出现在字符串的连字符(或简单地复制令牌，在令牌激活字段的第一个块定位光标，然后按 Ctrl + V 键)，如图 6-10 所示。

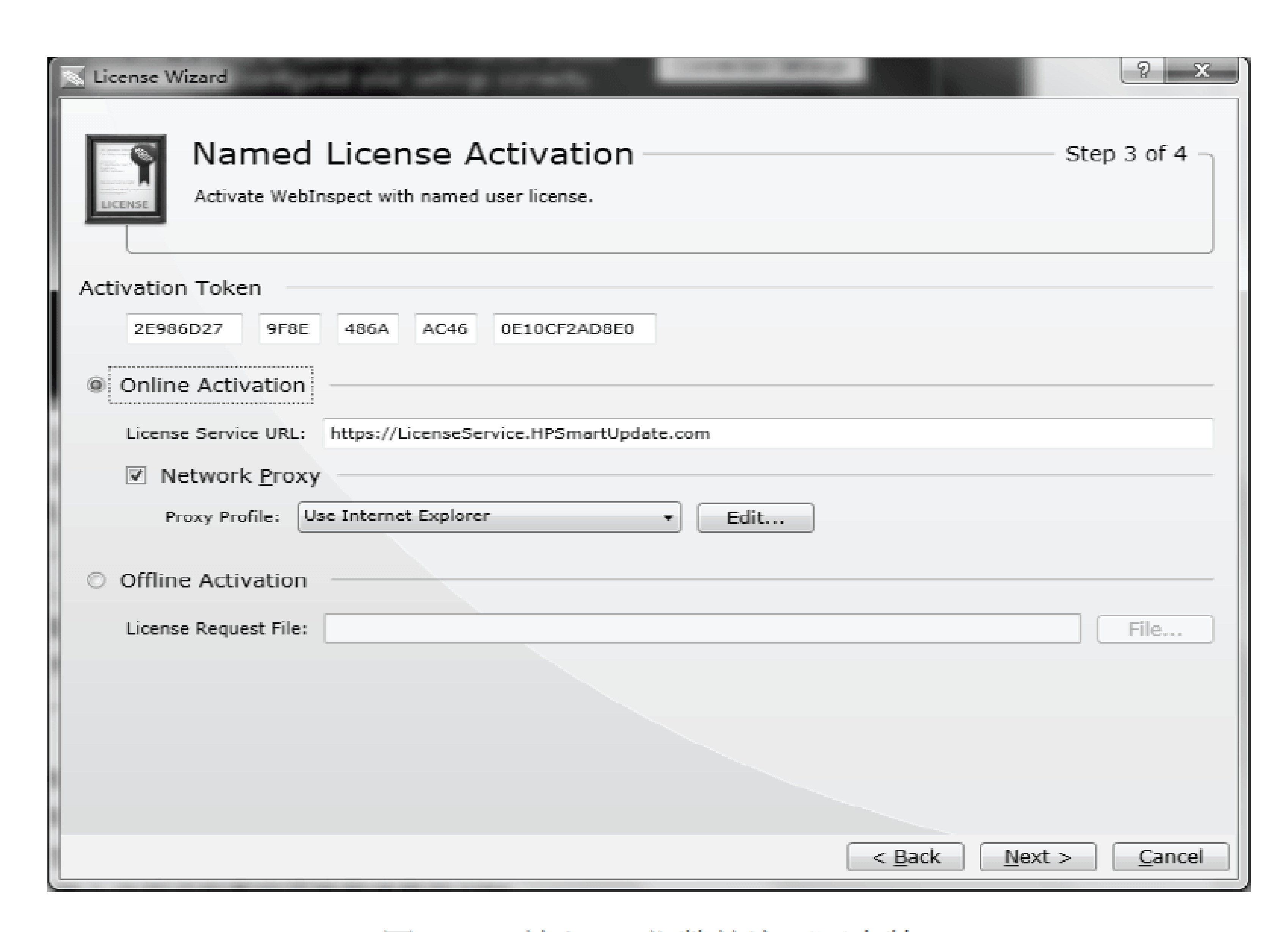



图 6-10　输入 32 位数的许可证令牌

6) 单击 Next(下一步)。关于所安装的许可证信息出现在授权范围部分，程序会显示用户的许可证信息并确认 WebInspect 的副本被激活，如图 6-11 所示。

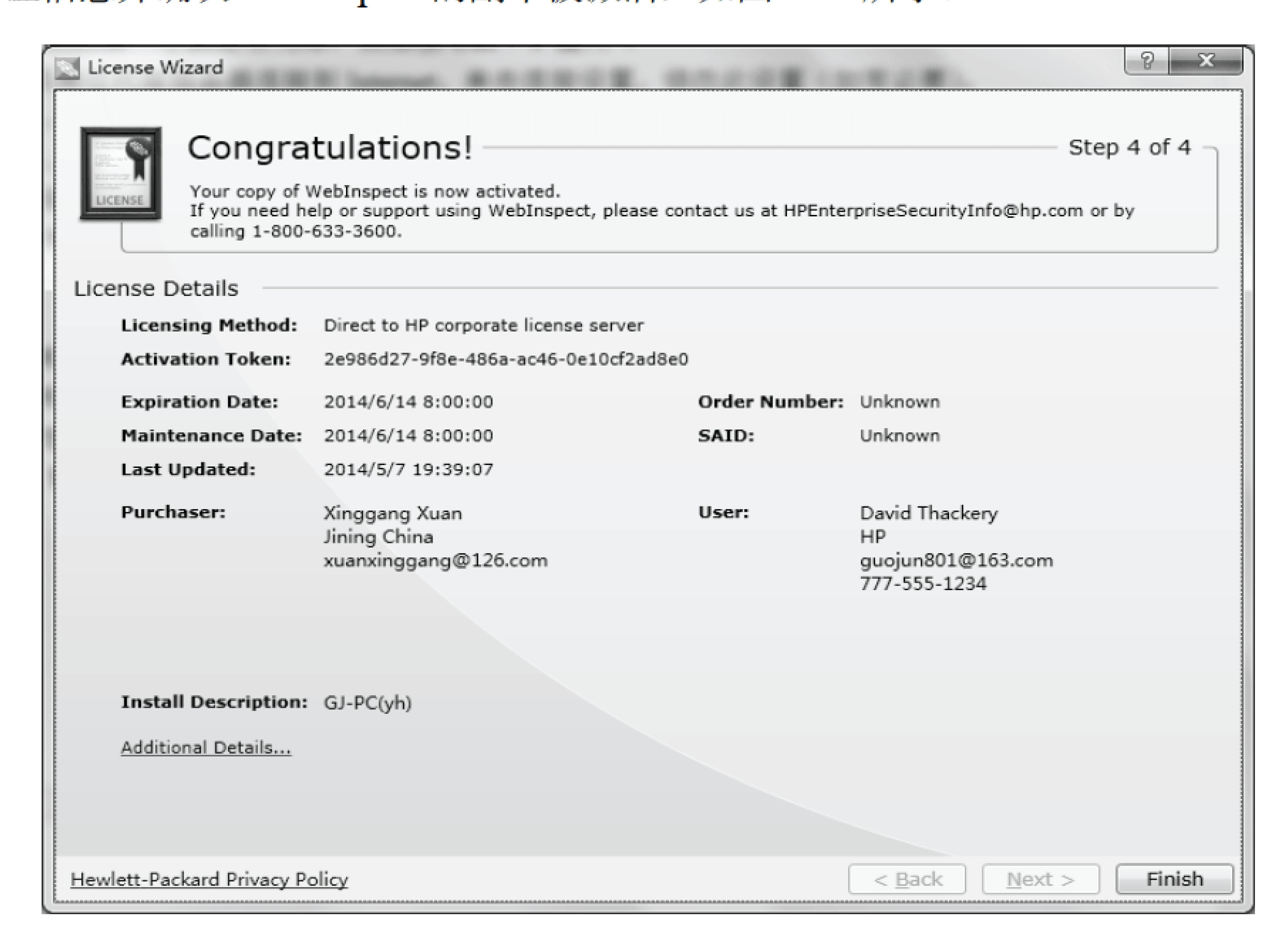



图 6-11　WebInspect 激活完成界面

7) 单击 Finish(完成)。

# 6.3　主要功能

- 图标方式：单击工具栏上代表相应命令的图标按钮；
- 自动执行 Web 应用程序安全测试和评估；
- 在整个生命周期中执行应用程序安全测试和协作；
- 通过最先进的用户界面轻松运行交互式扫描；
- 利用高级工具 (HP Security Toolkit) 执行渗透测试；
- 配置以支持任何 Web 应用程序环境。

# 6.4　本章小结

本章介绍了 HP WebInspect 的主要特征、新特征以及主要功能。HP WebInspect 是业内领先的网站应用安全评估解决方案，可以透彻分析复杂的 Web 应用程序和 Web 服务，检测出其中是否有安全漏洞。这套解决方案的技术涵盖面广泛、扫描功能速度快、漏洞说明与修复知识丰富，并能提供精确的网站扫描结果。在掌握这些必要的基本知识的同时，学会正确安装和激活 WebInspect 是扫描网站的前提。

# 第7章　WebInspect应用实践

## 本章导读

WebInspect 支持整个软件生存周期中对 Web 应用进行测试，同时高效管理测试结果。用户可以快速而准确地自动执行 Web 应用程序安全测试和 Web 服务安全测试，从而确保大部分易受攻击的输入点免遭攻击。WebInspect 安装也包括 HP 支持工具，可以帮助 HP 技术支持人员分析和解决用户在使用应用程序安全中心产品遇到的问题。本章介绍了 WebInspect 的新功能和各个按钮的用法和测试步骤，并简述了各个 HP 支持工具。

应掌握的知识要点：

- 了解 WebInspect 的新功能；
- 使用 WebInspect 的步骤；
- 了解 WebInspect 工具栏；
- 了解 WebInspect 菜单栏；
- 生成报告；
- 扫描网站；
- 向导扫描；
- 基础扫描；
- Web 服务扫描;
- 企业扫描；
- 了解 HP 支持工具。

## 7.1　WebInspect 10.20 的新功能和增强功能

惠普公司推出的 WebInspect 10.20 是目前的最新版本，较以往的版本，WebInspect 10.20 具有如下的新功能和增强功能。

### 1) 本地移动服务扫描

手动爬行本机移动应用程序并捕获网络流量作为工作流的宏。一旦捕获到从 Android 或 iOS 应用程序定向到后端服务的 HTTP 流量后，即可使用 WebInspect 的一套标准操作来进行重放、模糊测试和注入攻击等安全测试。这是一个专门添加到捆绑移动检查的新方法。

### 2) WebInspect 代理改进

WebInspect 代理(原名 SecurityScope)可与 WebInspect 在没有额外成本的条件下与用户界面和扫描引擎深度整合。当存在兼容性时，代理兼容性由预扫描分析器检测和通过链接进行

安装，并在用户界面呈现。该代理也采用一个新的活跃模式运作, 对于 WebInspect，也可以使用攻击策略去提高精度和性能。

#### 3) 7 种致命错误分类

由 Fortify Software 的安全研究小组与加里· 麦格劳博士(Dr. Gary McGraw)一起对软件开发的安全性进行了 7 种致命错误的分类。这种分类是为了帮助开发人员和安全人员理解常见的导致漏洞的编码错误类型。当构建和测试软件时，通过将这些错误形成一个简单的分类，开发人员可以很容易地识别导致安全漏洞的问题类别，并找出存在的错误。

#### 4) FIPS 的兼容性

WebInspect 配置符合(美国)联邦信息处理标准(FIPS，Federal Information Processing Standards)，可在 Windows 环境下运行。

#### 5) 通过浏览器模拟对移动网站的扫描

移动网站可在自定义用户代理下进行扫描，或在流行的移动平台用户代理下进行扫描。对于 Android 系统而言，Safari 或 Chrome 等浏览器都可供选择。在这种模式下，WebInspect 可以扫描该网站的内容，因为它将被渲染到移动浏览器中。

#### 6) WebInspect API(应用程序界面)

除了命令行工具，WebInspect 公开了 REST(表述性状态转移，Representational State Transfer)服务端点，以允许远程客户配置、控制和从扫描中检索数据。

#### 7) 显著改进了 JavaScript 引擎

- 通过升级到最新 Gecko 引擎，对 HTML5 和现代 Web 新标准提供更好的支持。
- 通过智能办法消除多余的执行脚本，使性能得以改善。

#### 8) 更新 Web 宏记录器

- 本地化 HTML5 记录支持；
- 改善准确性；
- 升级到最新的 Gecko 引擎。

#### 9) 改进的 HTML5 支持

DOM 使用 HTML5 特性操作客户端进行审计。

#### 10) 更新平台支持

- Windows 8
- Windows Server 2012
- IE 浏览器 10

#### 11) 许可证和基础设施管理器(LIM)通过 WebInspect 交付安装

LIM 不再分开购买。安装 WebInspect 后，LIM3.0 是驻留在文件系统中一个简单 Windows 安装程序文件。

12) 自动检测 CSRF 令牌和重新配置扫描设置

WebInspect 可以自动检测 CSRF 令牌和重新配置扫描设置。

13) BURP 导入

WebInspect 可在 WebProxy 工具中打开 BURP 文件，导入 BURP 流量作为工作流的宏。

14) 增加了对 GWT 的支持扫描

WebInspect 增加了对 GWT 的支持扫描。

# 7.2　使用 WebInspect

## 7.2.1　基本介绍

首次启动 WebInspect 时，应用程序在客户区显示 Start Page(开始页)选项卡，如图 7-1 所示。此选项卡显示超链接的五大主要功能：

- Start a Guided Scan(启动向导扫描)
- Start a Basic Scan(启动基础扫描)
- Start a Web Service Scan(启动 Web 服务扫描)
- Start an Enterprise Scan(启动企业扫描)
- Generate a Report(生成报告)
- Start SmartUpdate(启动智能升级)(非功能)

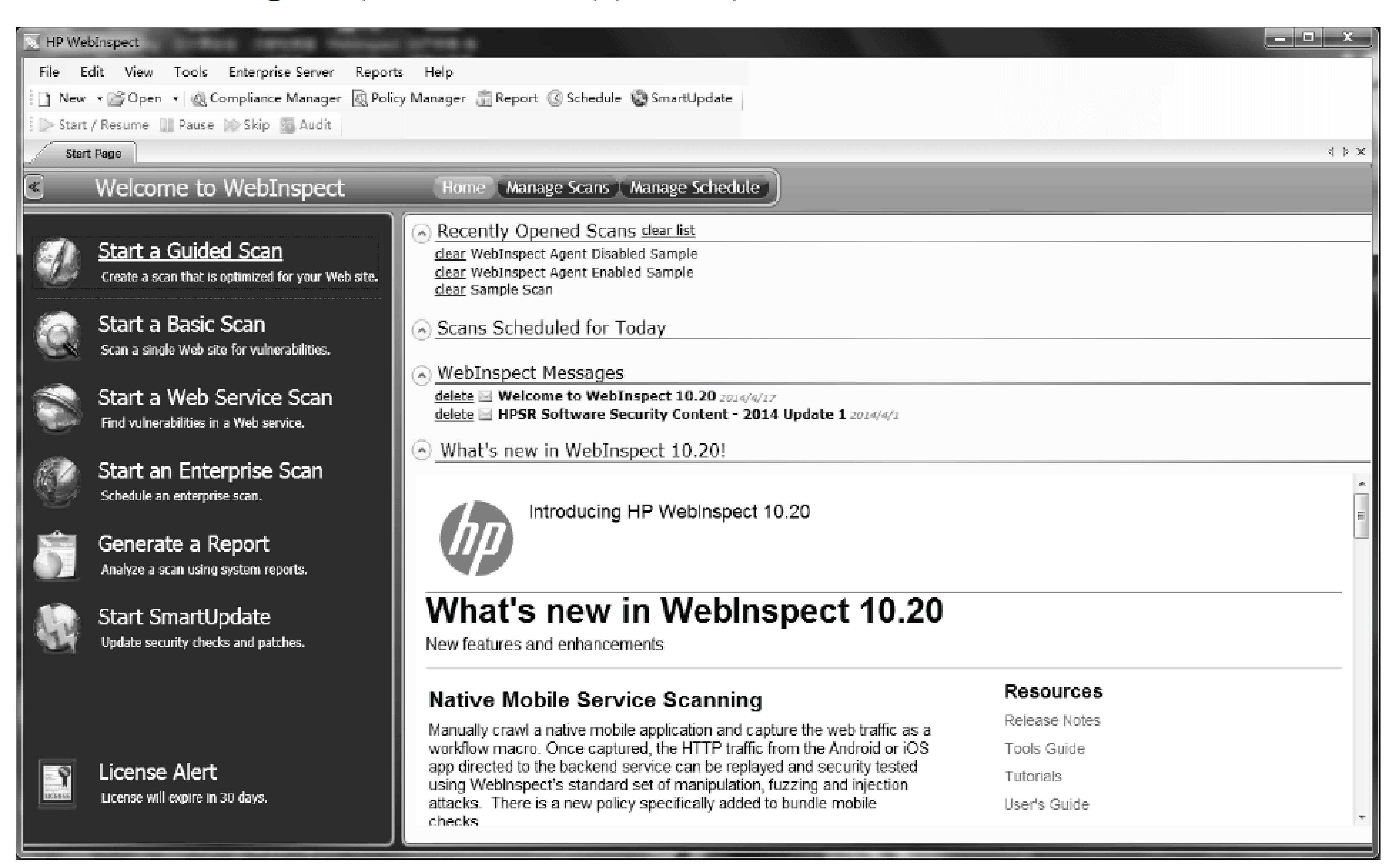



图 7-1　WebInspect 打开界面

用户可通过单击窗格上方条形栏的左箭头来关闭左窗格。右窗格中的内容是由按钮栏上的选择按钮确定，如图 7-2 所示。




图 7-2　按钮栏

按钮栏选项包括：

- Home(首页)：显示最近打开的扫描列表，今日进行的计划扫描，最近生成的报告，以及从惠普服务器下载的信息。
- Manage Scans(管理扫描)：显示先前进行的扫描列表，用户可以打开、重命名或删除列表。单击 Connections(连接)选择数据库：选择本地(扫描在用户机器上配置了 SQL Server 2008 Express 数据库列表)或远程(扫描在服务器上或网络上配置了 SQL Server 2008 Standard 的数据库列表)，或选择两者。
- Manage Schedule(管理计划)：显示预定要执行的扫描列表。用户可添加一个扫描计划表，编辑或删除一个预定的扫描，或手动启动扫描。

用户每次启动一个扫描，即 WebInspect 打开一个标记名称或目标网站描述的标签。这项工作区域被分为三个区域：导航窗格、信息窗格和摘要窗格，如图 7-3 所示。

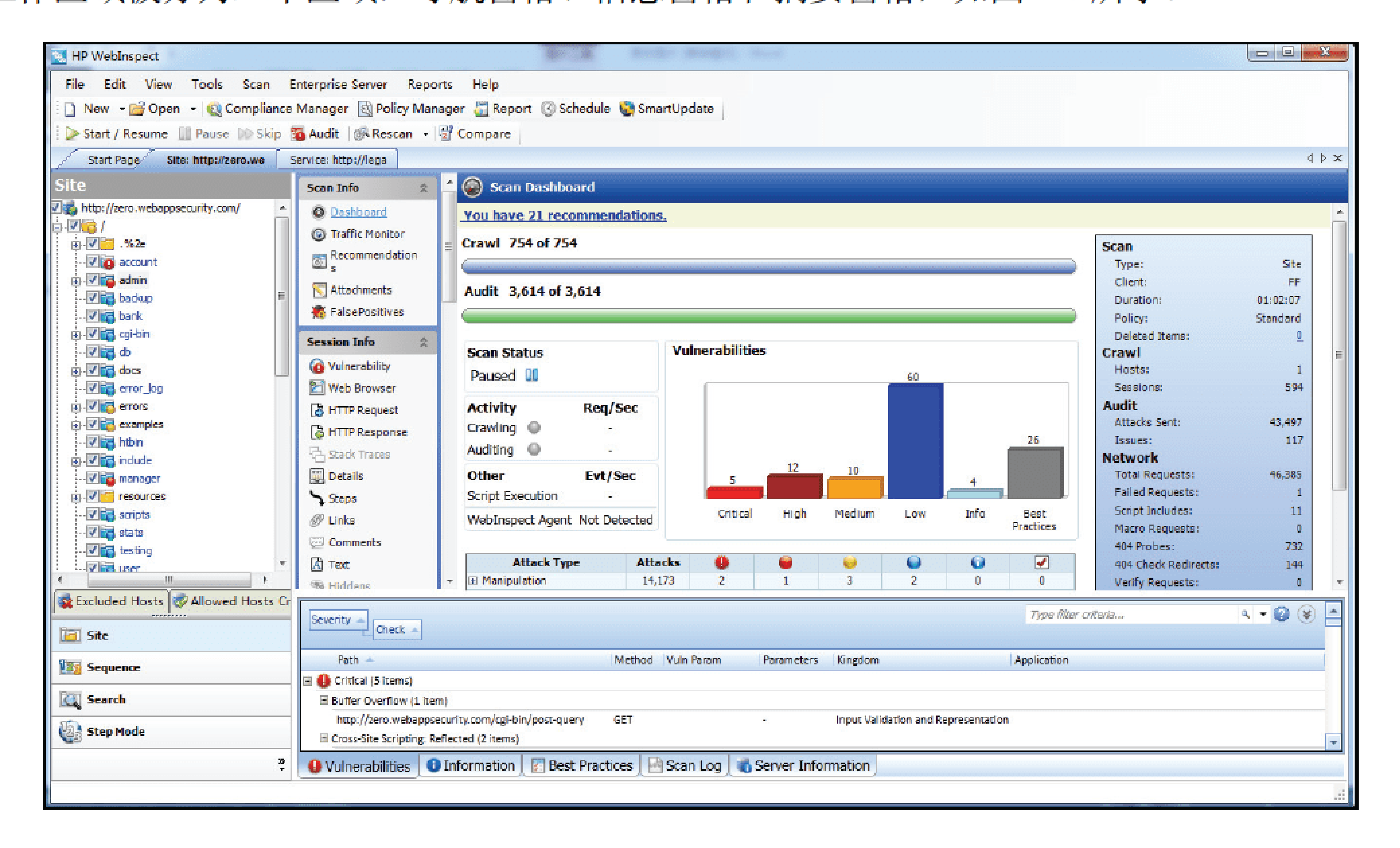



图 7-3　WebInspect 的三个工作区域

如果用户同时打开大量的扫描，导致没有足够的空间来显示所有标签，可以通过单击标签栏最右端的箭头◂▸来滚动标签。单击 X 关闭选定的标签。

## 7.2.2　导航窗格

在进行或查看扫描时，导航窗格是在WebInspect窗口的左侧。它包括 Site(网站)、Sequence (序列)、Search(搜索)以及 Step Mode (步模式)按钮，这些按钮用来确定内容(或“视图”)，如

图 7-4 所示。

在导航窗格底部的按钮分别对应不同的视图：网站、序列、搜索、步模式。如果未显示所有按钮，单击列表底部的下拉按钮，将显示更多按钮。

### 1. 网站视图

在导航窗格中，WebInspect 网站视图仅显示那些公布网站层次结构的会话，以及漏洞会被发现的会话。在网站进行爬行时，WebInspect 选择每个会话旁的复选框(默认情况下)，表明会话也将进行审计。当进行连续爬行和审计时(其中网站被审计之前完全进行爬行)，用户可通过清除相关的复选框，在审计开始前放弃对该会话进行审计。

网站视图还包含两个弹出选项卡：Excluded Hosts(排除主机)和 Allowed Hosts Criteria(允许主机标准)，如图 7-5 所示。

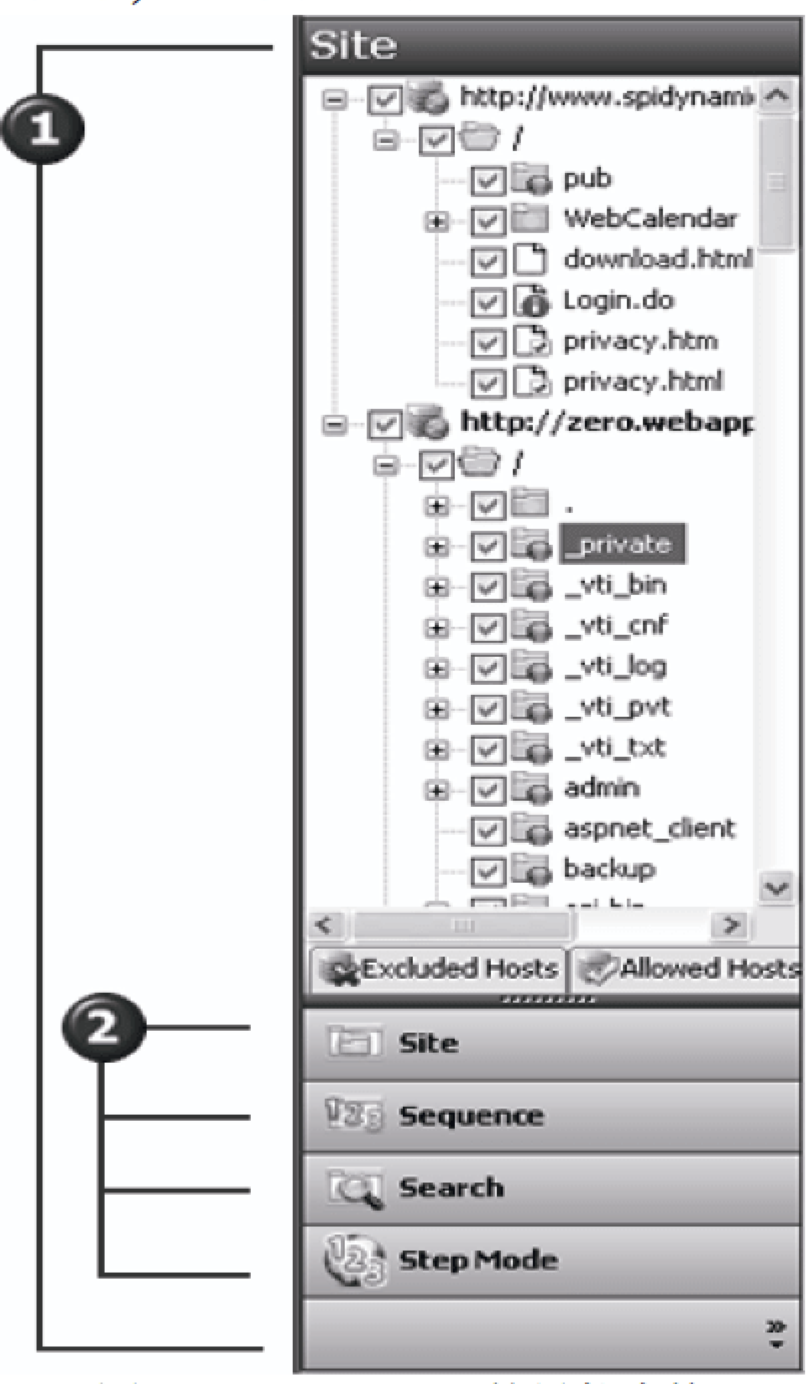



图 7-4　WebInspect 的导航窗格

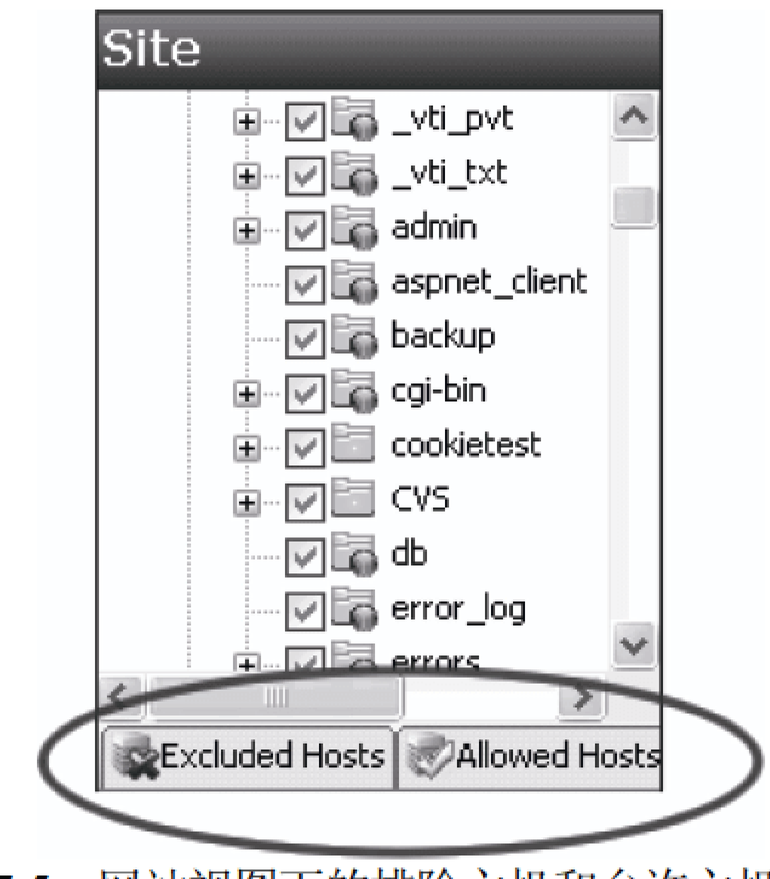



图 7-5　网站视图下的排除主机和允许主机标准

#### 1) 排除主机

如果单击 Excluded Hosts(排除主机)选项卡(或鼠标指针在其上面悬停)，该选项卡显示所有禁止的主机列表。这些主机可在目标网站内的任何地方被访问，但不能被扫描。要扫描这些主机，需要单击 Default/Current Scan Settings →Scan Settings→ Allowed Hosts(默认/当前扫描设置→扫描设置→允许主机)，即“允许主机设置”。

使用 Excluded Hosts(排除主机)选项卡，用户通过单击 Add to scan(添加到扫描)，或 Add allowed host criteria(添加到允许主机标准)，可选择要排除的主机。如图 7-6 所示。

Add to Scan 会在主机根目录下的网站树建立一个节点。WebInspect 将扫描该节点。如果用户选择 Default/Current Scan Settings →Scan Settings →Session Storage(默认/当前扫描设置→扫描设置→会话存储)，WebInspect 将扫描整个主机。

在当前扫描设置中，“添加到允许主机标准”，即添加 URL 到允许主机列表中，WebInspect

将扫描指向该主机中的所有后续链接。但当 WebInspect 已经扫描指向该主机上包含一个链接的唯一资源后，如果用户在“允许主机标准”上添加一个主机，新增的主机将不会被扫描。

### 2) 允许主机标准

如果单击 Allowed Hosts Criteria(允许主机标准)选项卡(或鼠标指针在其上面悬停)，则会在标签中显示指定 WebInspect 扫描设置(在允许主机)的 URL(或正则表达式)。

如果单击 Delete(删除)或 Add allowed host criteria(添加允许主机标准)，WebInspect 可以打开当前设置对话框，用户可以添加、编辑或删除允许主机标准(文字的 URL 或正则表达式表示的 URL)，如图 7-7 所示。




图 7-6　排除主机选项卡　　　　图 7-7　允许主机标准选项卡

如果用户添加一个条目，WebInspect 将扫描后续环节遇到的符合标准的任何主机。然而在 WebInspect 扫描指向该主机包含一个链接的唯一资源后，如果用户再指定一个主机，新增的主机将不会被扫描。同样，如果用户删除了允许主机列表中的条目，仍将扫描 WebInspect 遇到的任何资源。

如果要保存这些设置便于将来扫描，选择 Save settings as(保存设置)(在设置窗口左侧窗格的底部)。

对“排除主机”或“允许主机标准”进行修改前，必须暂停扫描。此外，增加或删除主机扫描是否可用，取决于用户暂停的扫描点。例如，WebInspect 已经扫描指向添加主机包含一个链接的唯一资源后，如果用户添加一个允许主机，则新增的主机将不会被扫描。

## 2. 序列视图

顺序图显示了 WebInspect 进行扫描过程中服务器资源的显示顺序。

注意：在网站视图和序列视图中，蓝色文本表示目录或文件，由 WebInspect “猜”到的(而不是通过链接发现的)资源。例如，如果目标网站包含一个名为“备份”的文件，WebInspect 便会不断提交请求“GET/备份/ HTTP/1.1”，试图去找到它。

## 3. 搜索视图

搜索视图允许用户在各种 HTTP 消息组件的所有会话中进行搜索。例如，如果从下拉列

表中选择 Response Raw 并指定“set-cookie”作为搜索字符串，WebInspect 便列出每一个会话的原始 HTTP 响应中包含的“set-cookie”命令，如图 7-8 所示。

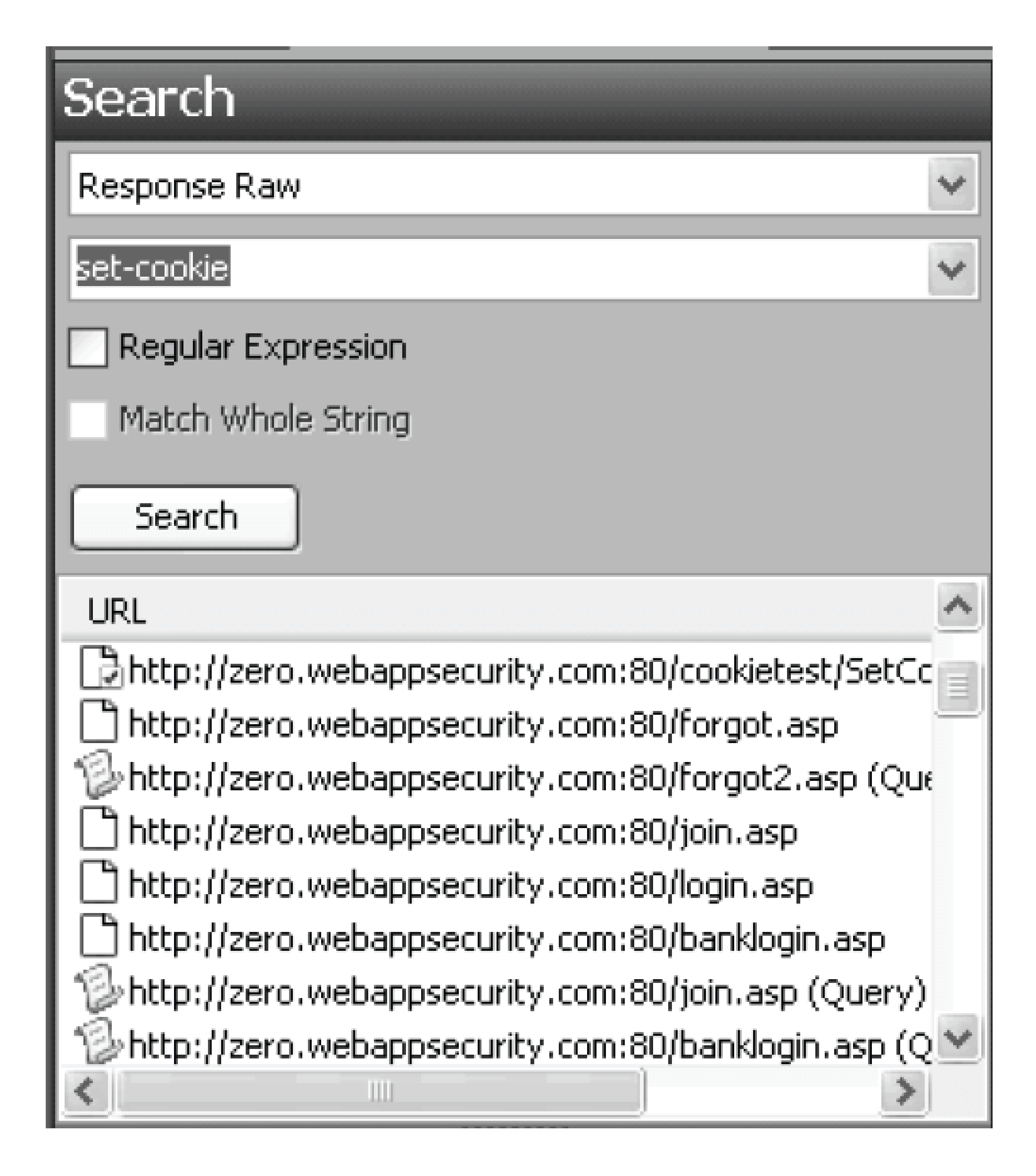



图 7-8　搜索视图

如果要使用搜索视图：

1) 在导航窗格中，单击 Search(搜索)(在面板的底部)。

2) 从最上面的列表中，选择要搜索的区域。选项包括：

- Status Code(状态代码)
- URL(网址)
- Request Raw(请求原)
- Request Method(请求方法)
- Request Post Data(请求发送数据)
- Request Post Data Name(请求发送数据名称)
- Request Post Data Value(请求发送数据值)
- Request Headers(请求头)
- Request Header Name(请求头名称)
- Request Header Value(请求头值)
- Request Query(请求查询)
- Request Query Name(请求查询名称)
- Request Query Value(请求查询值)
- Request Cookies(请求的 Cookie)
- Request Cookie Name(请求的 Cookie 名称)
- Request Cookie Value(请求的 Cookie 值)
- Request File Name and Extension(请求文件名和扩展名)
- Request File Name(请求文件名)

- Request File Extension(请求文件扩展名)
- Request Path(请求路径)
- Response Raw(响应原)
- Response Headers(响应头)
- Response Header Name(响应头名称)
- Response Header Value(响应标头值)
- Response Cookies(响应的 Cookie)
- Response Cookie Name(响应的 Cookie 名称)
- Response Cookie Value(响应的 Cookie 值)

3) 在组合框中，键入或选择要查找的字符串。

4) 如果该字符串代表一个正则表达式，则选择 Regular Expression(正则表达式)复选框。

5) 在 HTTP 消息中要查找完全匹配的整个字符串，需选择 Match Whole String(匹配整个字符串)复选框，精确匹配不区分大小写。注意：该选项不适用于某些搜索目标。

6) 单击 Search(搜索)。

### 4. 步模式

使用步模式手动浏览整个网站，从网站视图或序列视图中选择一个会话开始。

按照下面的步骤逐步浏览网站：

1) 在网站视图或序列视图中，选择一个会话。

2) 单击 Step Mode(步模式)按钮。如果按钮不可见，请单击 Configure Buttons(配置按钮)的下拉菜单，然后选择 Show More Buttons(显示更多按钮)。

3) 当步模式显示在导航窗格中，从 Audit Mode 列表选择 Audit as you browse(浏览审计)或 Manual Audit(手动审计)。建议选择后者，如图 7-9 所示。

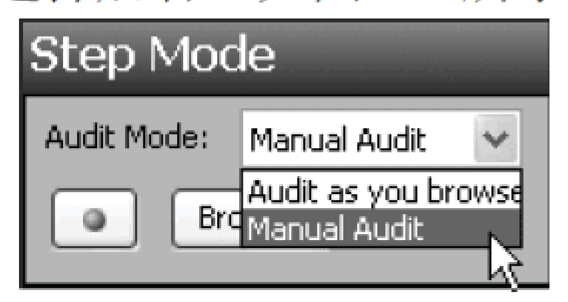



图 7-9　步模式中选择手动审计

4) 单击 Record(记录)。

5) 单击 Browse(浏览)。

浏览器会开启并显示与所选会话相关联的响应。继续浏览尽可能多的页面。

6) 完成后，返回 WebInspect 并单击 Finish(完成)。新的会话被添加到导航窗格中。

7) 如果在步骤 3 中选择 Manual Audit(手动审计)，单击 Audit。WebInspect 将审计所有未经审计的会话，包括用户通过步模式添加(或更换)的审计。

### 5. 导航窗格中的图标

使用表 7-1 和表 7-2 来识别序列视图和网站视图中显示的资源。

表 7-1　在导航窗格使用的图标

| 图　标 | 定　义 |
|---|---|
| | 服务器/主机：代表网站树状结构的顶层 |
| | 蓝色文件夹：由 WebInspect 发现的用户 Web 服务器上的私人文件夹。这些文件夹和网站本身没有关系 |
| | 黄色文件夹：文件夹的内容在网站上都可以用 |
| | 灰色文件夹：用于指明该条目通过路径截断发现一个文件夹。一旦父文件夹(上一级文件夹)被发现，它的属性决定了该文件夹将显示为蓝色或黄色 |
| | 文件 |
| | 请求或发送 |
| | DOM 事件 |

表 7-2　图标叠加在一个文件夹或文件上表示发现漏洞

| 图　标 | 定　义 |
|---|---|
| | 一个红色的惊叹号表示该对象包含一个严重漏洞。攻击者可在服务器上执行命令，如检索或修改个人信息 |
| | 红点表示的对象包含一个高级漏洞。一般来说，能够查看源代码、文件的 Web 根目录和敏感的错误消息 |
| | 金点表示该对象包含一个中等漏洞。这些都是一般的敏感的非 HTML 错误或问题 |
| | 蓝点表示对象包含低等漏洞。这些一般都是需要关注的问题，或可能成为更高的漏洞 |
| | 在一个蓝色圆圈中的“i”表示一个信息项。这些都是网站中需要关注的点，或特定的应用程序或 Web 服务器 |
| | 红色的复选标记表明违反“最佳实践(Best Practice)” |

每个对象都代表一个会话，这是一个匹配的集合，包括：WebInspect 发送 HTTP 请求去检测漏洞以及来自服务器的 HTTP 响应。

### 6. 导航窗格中的快捷菜单

右键单击会话(在导航窗格中的条目)，显示表 7-3 中所描述命令的快捷菜单。

表 7-3　导航窗格快捷命令

| 命　令 | 定　义 |
|---|---|
| Expand Children(自动扩展子级) | 展开分支网站中的树节点(仅适用于网站视图) |
| Collapse Children *(折叠子集) | 合并分支网站到上级节点(仅适用于网站视图) |
| Check All*(查看全部) | 标记所有子集的复选框(仅适用于网站视图) |

(续表)

| 命　　令 | 定　　义 |
| --- | --- |
| UnCheck All*(全部取消) | 从所有子集中删除复选标记(仅适用于网站视图) |
| Generate Session * Report* (生成会话报告) | 创建一个报告，显示摘要信息、攻击请求和攻击响应、链接和网址、注释、表格以及电子邮件地址，并为所选会话检查说明(仅适用于网站视图) |
| Export Site Tree * (导出网站树) | XML 格式的网站树保存到用户具体指定的某个位置(仅适用于网站视图) |
| Copy URL(复制网址) | 将选定会话的 URL 复制到剪贴板，选择 Edit→Copy URL(编辑→复制 URL) |
| View in Browser (浏览器视图) | 在浏览器中显示 Web 服务器的响应 |
| Links(链接) | 列出目标网站的所有资源，包括选定资源的链接(仅适用于网站视图)。这些链接可以通过 HTML 标签、脚本或 HTML 形式呈现。它还列出由选定会话的 HTTP 响应中的链接所引用的所有资源<br>如果用户双击一个链接，WebInspect 可将焦点转移到导航窗格中所引用的会话。或者，也可以在 Web 浏览器中通过查看会话浏览该链接资源(单击 Web Browser) |
| Add(添加) | 允许添加 WebInspect 扫描的其他地点发现(人工检查，其他工具等)以供参考。此外，用户可以添加漏洞位置，对网站进行完整分析 |
| Edit Vulnerability(编辑漏洞) | 允许编辑被手动添加的漏洞或编辑漏洞的位置 |
| Remove Location (删除位置) | 从导航窗格(包括网站和序列视图)删除选定会话，并删除任何相关漏洞。注意：用户可以恢复删除位置(会话)及其相关的漏洞 |
| Crawl(爬行) | 重新爬行选定的 URL |
| Review Vulnerability (审阅漏洞) | 允许用户重新测试该漏洞，将其标记为“忽略”或“误报”，或发送到 HP 质量中心或 IBM Rational ClearQuest |
| Mark as False Positive (误报标记) | 标记漏洞作为误报，并允许用户添加注释 |
| Remove Server (删除服务器) | 从导航窗格中删除服务器，不包括任何剩余扫描活动的服务器。当右键单击服务器时，此命令才会显示 |
| Attachments (附件) | 允许创建与选定会话相关的说明，标志的会话跟进，或增加一个漏洞说明，或添加一个漏洞快照 |
| Tools(工具) | 介绍可用工具的子菜单 |
| Filter by Current Session (当前会话过滤器) | 过滤摘要窗格所选会话的摘要数据 ID 条目 |
| Send to(发送到) | 对于一个缺陷，允许用户转换选定漏洞，并发送到 HP 质量中心或 IBM Rational ClearQuest，使用了 WebInspect 应用程序设置中指定的配置文件 |

备注：只有在导航窗格使用本网站视图时，*命令才显示在快捷菜单上。

### 7.2.3　信息窗格

在进行或查看扫描时，信息窗格包含三个可折叠的信息板和一个信息显示区。1 代表扫

描信息面板(Scan Info Panel)，2 代表会话信息面板(Session Info Panel)，3 代表主机信息面板(Host Info Panel)，4 代表信息显示区(Information Display Area)，如图 7-10 所示。

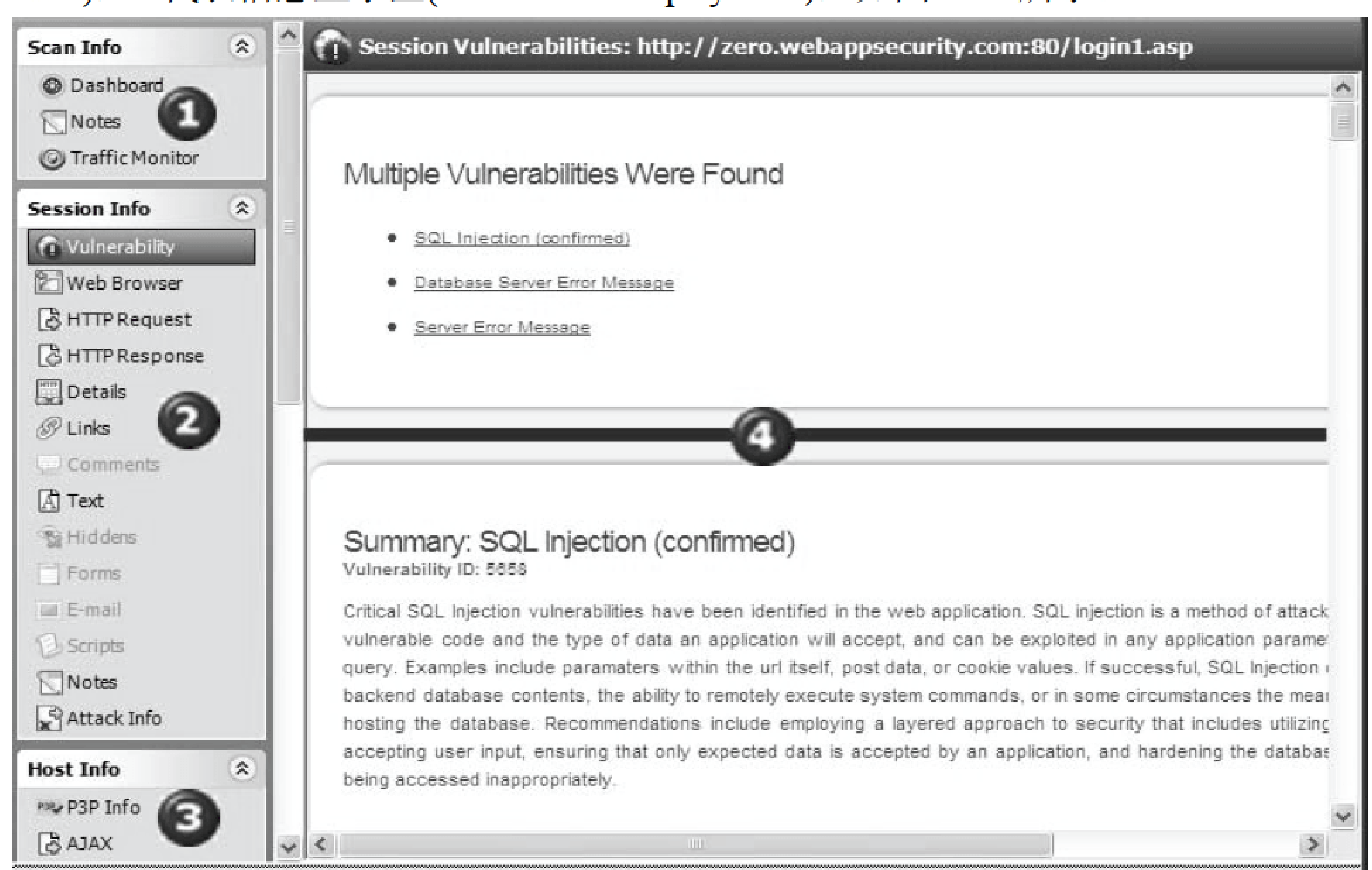



图 7-10　信息窗格的三个可折叠的信息板和一个信息显示区

通过单击左栏中三个信息面板之一的某个条目选择显示信息类型。

**注意**　如果用户查看该漏洞信息，打开链接后，需在导航窗格中单击 highlighted session (突出显示会话)来返回。

### 1. 扫描信息面板

Scan Info(扫描信息)面板有五个默认选项：Dashboard(仪表板)、Traffic Monitor(流量监控器)、Recommendations(注意事项)、Attachments(附件)和 False Positives(误报)。

仪表板显示对实时扫描结果的汇总和用图形表示的扫描进展，如图 7-11 所示。

Crawl Gauge(爬行统计)：爬行会话数/爬行会话总数。

Audit Gauge(审计统计)：审计会话数/审计会话总数。

Scan Status(扫描状态)：状态包括正在运行，暂停或完成。

Crawling Req/Sec(爬行请求/秒)：爬行当前网络运行的每秒请求数(运行步长为 5 秒)。

Auditing Req/Sec(审计请求/秒)：审计当前网络运行的每秒请求数(运行步长为 5 秒)。

Script Evt/Sec(脚本事件/秒)：正在被执行的脚本事件数(运行步长为 5 秒)。

Vulnerability Graph(漏洞走势图)：各类级别中已确定为问题的总数。

Attack Stats Grid(攻击数据网格)：统计攻击和发现的问题，以及攻击类型和审计引擎的数量。

Duration(时间)：扫描已经运行的时间长度(如果扫描异常终止，可能不正确)。

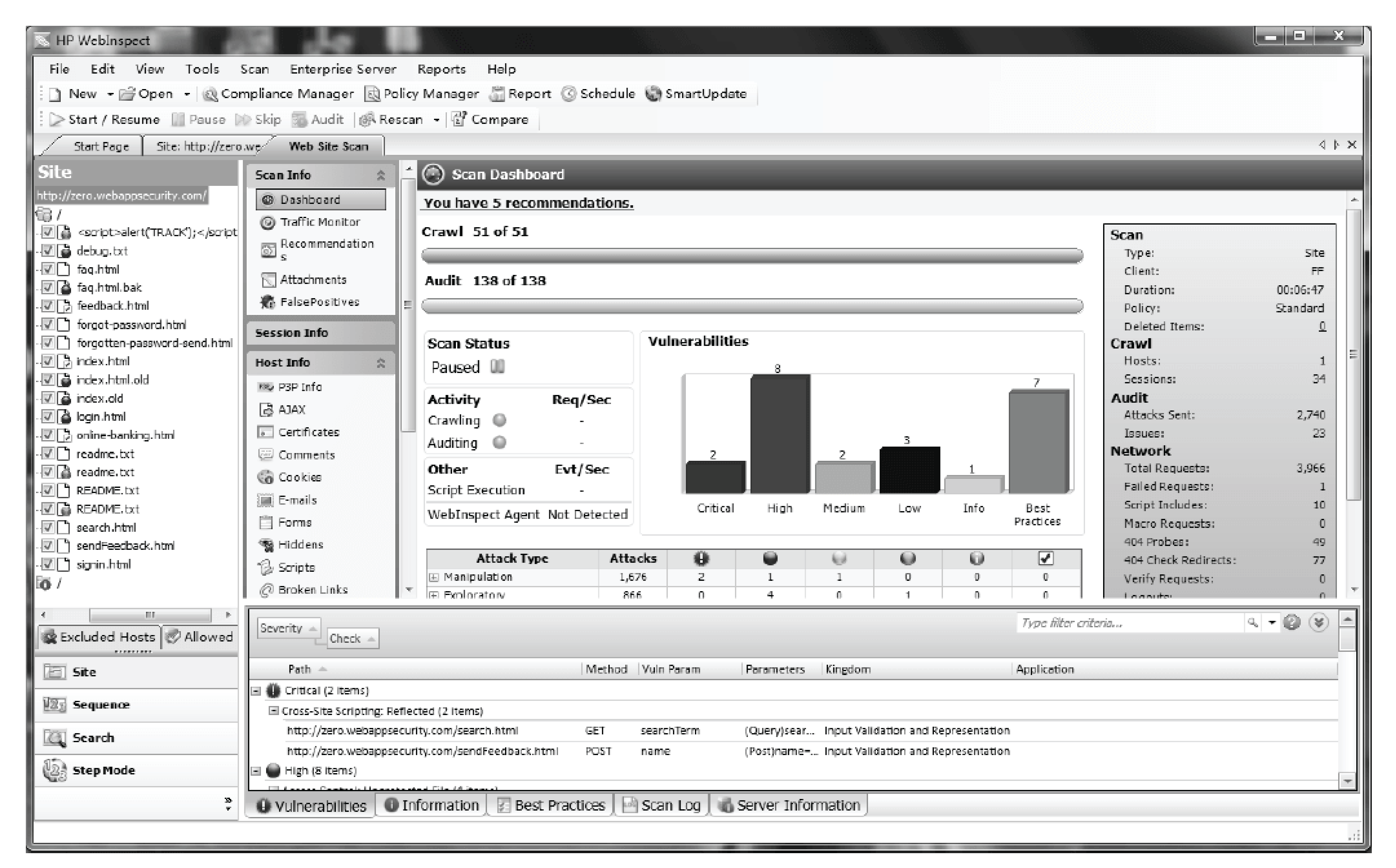


图 7-11　仪表板

Policy(政策)：用于扫描策略的名称。

Hosts(主机)：包括正在扫描的主机数。

Sessions(会话)：会话总数(不包括 AJAX 请求，包括脚本和脚本框架，包含 WSDL)。

Attacks Sent(攻击发送)：发送攻击总数。

Issues(问题)：问题发现总数(所有漏洞，以及最佳实践)。

Total Requests(总要求)：提出请求的总数。

Failed Requests(失败请求)：失败的请求总数。

Script Includes(脚本包括)：包括脚本的总数。

Macro Requests(宏请求)：宏请求数。

404 Probes(404 探头)：文件未找到探测器的数量来确定文件未找到状态。

404 Check Redirects(404 重定向检查)：导致重定向 404 探头的次数。

Verify Requests(验证请求)：检测存储参数的请求。

Logouts(注销)：注销检测次数和登录宏执行。

Macro Playbacks(宏回放)：已经执行宏的次数。

AJAX Requests(AJAX 请求)：AJAX 请求总数。

Kilobytes Sent(发送千字节总数)：发送千字节总数。

Kilobytes Received(接收千字节总数)：收到千字节总数。

注释：指出选择显示所有注释的列表，每一个都与特定的会话相关联。双击一个报告显示其内容，如图 7-12 所示。

在导航窗格中，WebInspect 通常只显示那些揭示网站层次结构的会话，以及漏洞会被发现的会话。流量监控器选择允许显示和评估每个 HTTP 请求发送的 WebInspect，和从服务器接收到 HTTP 相关的响应。要显示此选项，则必须选择 Enable Traffic Monitor Logging(启用流

量监控日志)，在默认设置(单击 Edit→Default Settings(编辑→默认设置)，然后，在扫描设置类别中，选择 General(一般)，如图 7-13 所示。

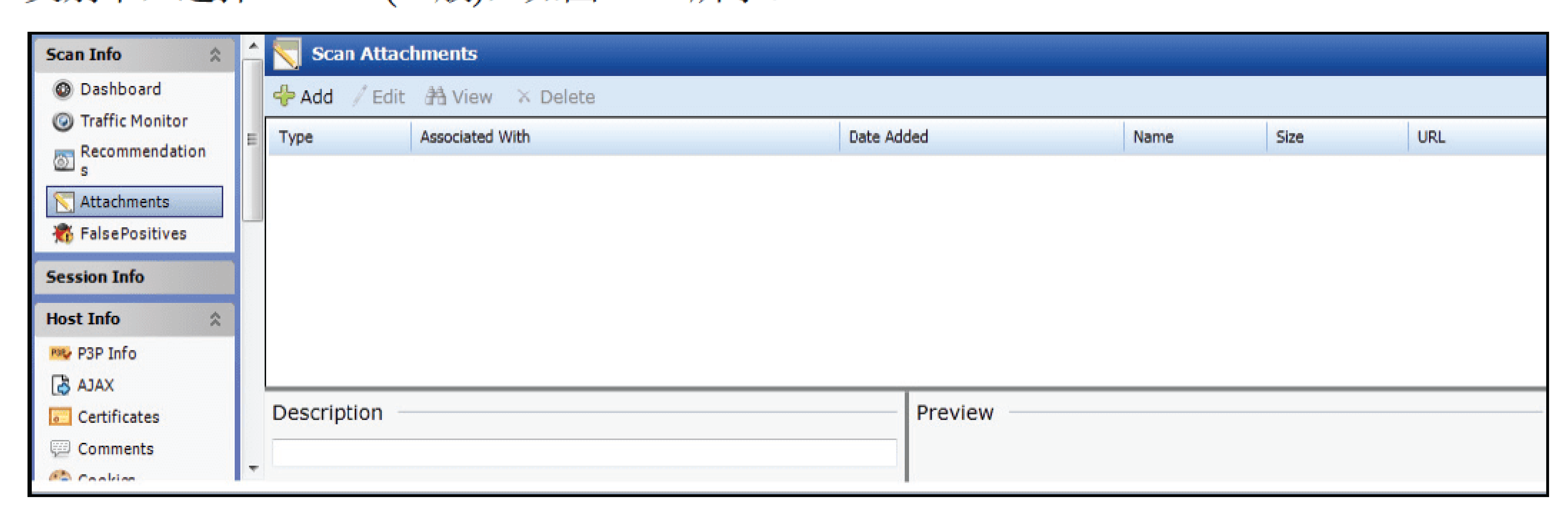



图 7-12　注释列表

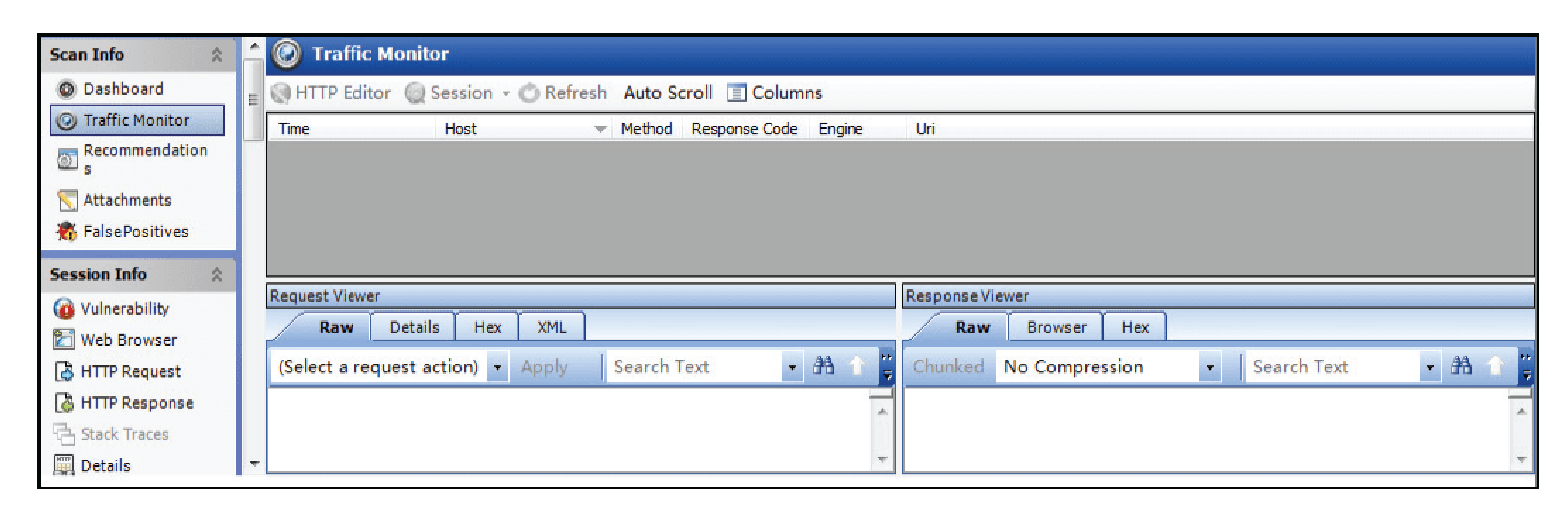



图 7-13　流量监控

该附件选项显示所有已添加到扫描列表的会话说明、漏洞说明、随访标志和漏洞截图。每个附件都与一个特定的会话相关联。此外还列出了扫描说明(说明适用于整个扫描，而不是某个特定的会话)。

用户可以创建一个扫描说明，或者编辑或删除现有的附件。

查看附件。请选择附件，然后单击 View(视图)(或直接双击附件)。

创建一个扫描说明。单击 Add(添加)菜单(在信息显示区)。

编辑附件。选择附件，然后单击 Edit(编辑)。注意，截图不能编辑。

可通过右键单击 Attachments(附件)，然后从快捷菜单中选择对应选项。也可以选择 Go to session(前往会话)，这将打开会话信息附件窗格，选择与该附件相关联的会话。

要选择在 WebInspect 用户界面的其他区域创建的附件，可右键单击导航窗格中的会话，然后从快捷菜单中选择附件，或右键单击摘要窗格中漏洞选项卡上的网址，然后从快捷菜单中选择附件。

此功能列出 WebInspect 原本标记为漏洞的所有 URL，但后来用户确定有些 URL 是误报。也可将先前的分析作为误报的扫描漏洞列表进行导入。然后在当前扫描中用 WebInspect 关联这些从先前漏洞检测到的扫描，并标出新出现的误报。

1) 从扫描信息面板中选择 False Positives(误报)，如图 7-14 所示。

2) 如有必要，单击漏洞描述的加号，以显示相关的 URL 和状态。

3) 单击一个 URL 查看评论(位于信息窗格的底部)。

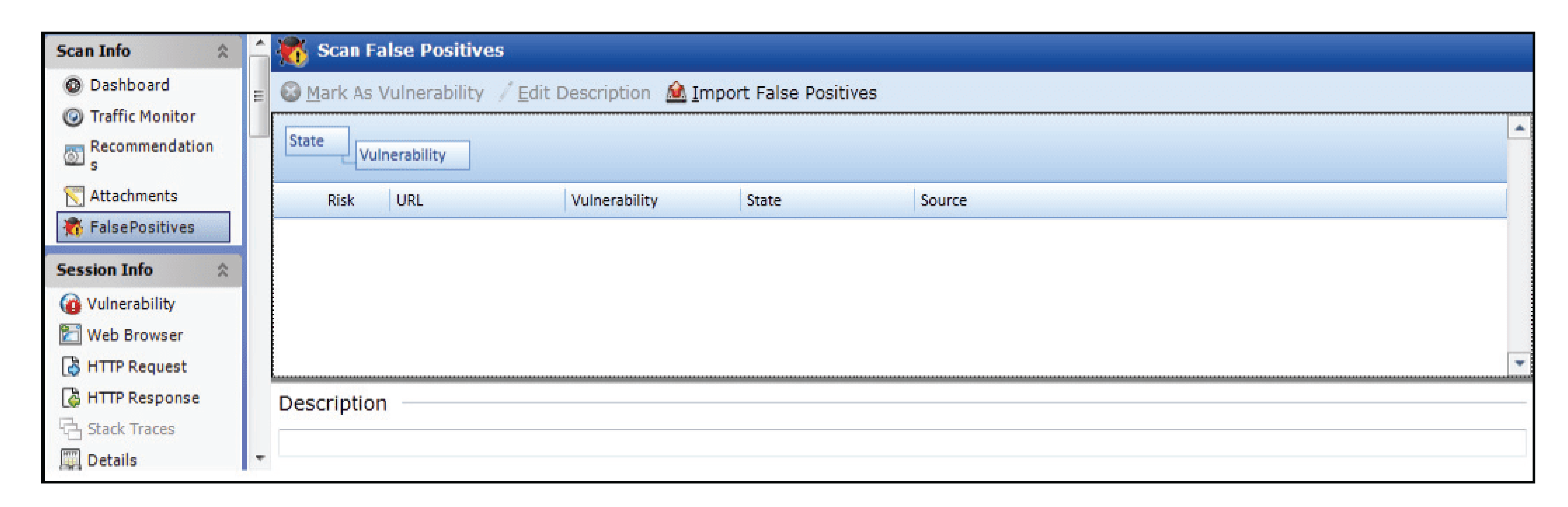



图 7-14　误报

4) 要从其他扫描导入误报，单击 Import Scans(导入扫描)。

5) 要更改误报并返回一个漏洞，从活动误报列表中选择一个条目，然后单击 Mark as Vulnerability(标记为漏洞)。

6) 要从暂无误报列表中删除某个条目，选择该条目，然后单击 Remove From Inactive(非活动删除)。

7) 要编辑与误报关联的注释，选择该条目，然后单击 Edit Comment(编辑评论)。

## 2. 会话信息面板

在导航窗格使用网站视图或序列视图，WebInspect 列出每个会话期间创建的扫描。选择一个会话，然后单击会话信息面板中的选项之一，以显示有关会话相关的信息，如图 7-15 所示。

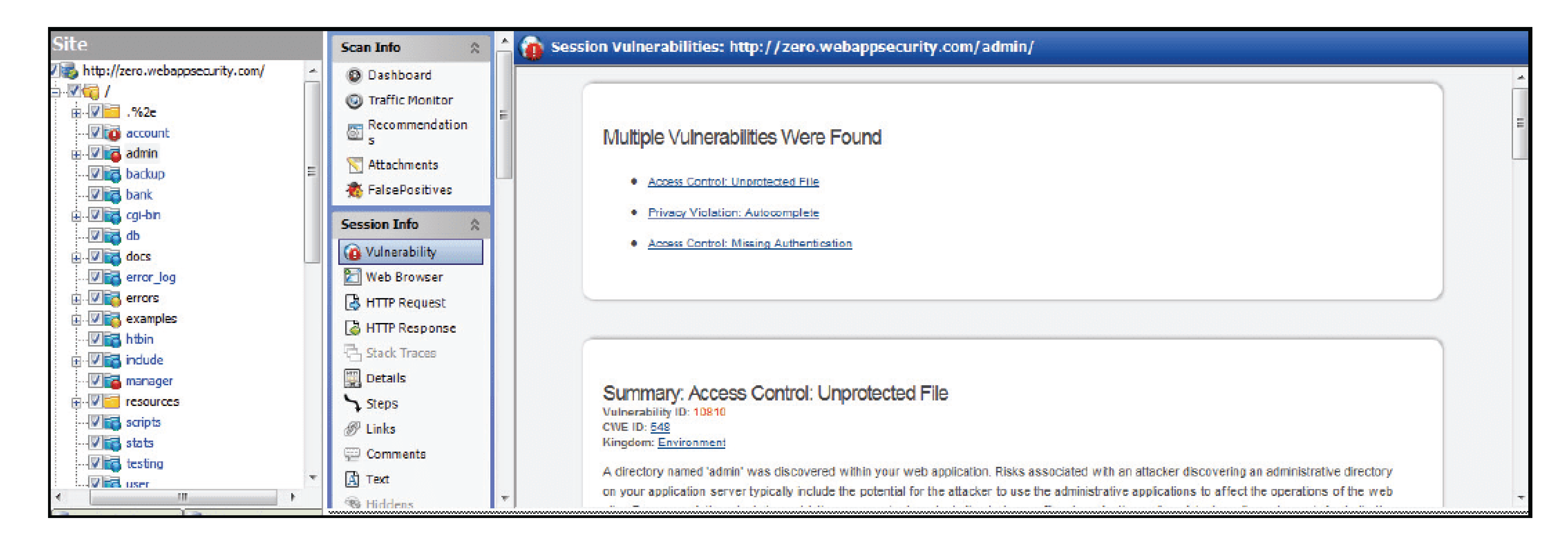



图 7-15　会话信息面板

下面例子的扫描中，WebInspect 发送 HTTP 请求 GET/stats/ HTTP/1.1。要查看 HTTP 响应，可执行以下步骤。

1) 在导航窗格中选择Stats.html。

2) 单击会话信息面板中的 Vulnerability(漏洞)。

表7-4列出了在会话信息面板中可用的选项。有些选项只出现在特定的扫描(向导扫描、基础扫描或Web服务扫描)中。此外，当且仅当它们是相关的所选会话，才能启用选项；例如，如果会话不包含表单，Forms(表单)选项不可用。

表 7-4　会话信息面板选项

| 选　项 | 定　义 |
|---|---|
| Vulnerability (漏洞) | 显示导航窗格中所选会话的漏洞信息 |
| Web Browser (Web 浏览器) | 在导航窗格中，通过 Web 浏览器显示服务器对所选会话的响应 |
| HTTP Request (HTTP 请求) | 显示由 WebInspect 发送到主机要扫描的目标网站服务器的原始 HTTP 请求 |
| HTTP Response (HTTP 响应) | 显示服务器原始 HTTP 响应的 WebInspect 请求。如果响应包含一个或多个攻击签名(表示一个漏洞被发现)，可单击按钮，标记从一个攻击信号到下一个攻击信号。<br>注意：如果选择一个 Flash(SWF)文件，WebInspect 可以显示 HTML，而不是二进制数据。这使得 WebInspec 以可读格式显示链接 |
| Stack Traces (堆栈跟踪) | 此功能的设计，可支持 HP Fortify SecurityScope 安装和运行在目标服务器上。对于某些检查(如 SQL 注入、命令执行和跨网站脚本)，SecurityScope 拦截 WebIns pect 的HTTP 请求，并在目标模块运行时进行分析。如果这个分析证实存在一个漏洞，SecurityScope 追加堆栈来跟踪 HTTP 响应。开发人员可分析这个堆栈跟踪来调查需要修复的区域 |
| Details(详细信息) | 显示请求及响应的详细信息列表，如响应程度和请求方法 |
| Comments(评论) | 显示所有嵌入在 HTTP 响应中的评论 |
| Steps (步骤) | 在导航窗格或选择的 URL 摘要窗格中，显示采用 WebInspect 到达选定会话的路线。从父会话(上一级会话)开始，该序列包括了后续访问的 URL，并提供了有关扫描方法的细节 |
| Links (链接) | 列出(在链接下)目标网站包含的全部所选链接资源。可以通过 HTML 标签、脚本或 HTML 形式呈现这些链接。它还列出由所选会话的 HTTP 响应的链接所引用的所有资源 |
| Text(文本) | 显示包含在 HTTP 响应中的所有文本 |
| Hiddens(隐藏) | 显示每个输入元素控制类型的名称是“隐藏” |
| Forms(表格) | 显示通过浏览器来呈现 HTML 形式的解释 |
| E-mail(电子邮件) | 显示包含在响应中的所有电子邮件地址 |
| Scripts(脚本) | 显示嵌入在服务器响应中的所有客户端脚本 |
| Attack Info (攻击信息) | 显示攻击序列号码、URL、审计引擎的名称以及脆弱性(漏洞)试验的结果。如果选择重新检查，WebInspect 可启动 HTTP 编辑器，允许用户重新发送相关的 HTTP 请求。进行扫描时，此功能是最常用的，以确定是否该漏洞已经得到修正。<br>攻击信息通常与创造该攻击的会话(而不是检测到的会话)相关联。如果攻击信息没有出现所选脆弱的阶段，选择上一级会话 |

(续表)

| 选　项 | 定　义 |
| --- | --- |
| Attachment<br>(附件) | 显示所有注释、标志和与所选对象相关联的截图。<br>要创建注释，用户可以：<br>•右键单击导航窗格中的会话，然后从弹出的菜单中选择注释<br>•右键单击摘要窗格中的漏洞选项卡上的 URL，然后从弹出菜单中选择注释。<br>•首先选择导航窗格中的会话，然后选择会话信息面板注释，继而单击添加菜单(在信息窗格中)。<br>只要向惠普质量中心或 IBM Rational ClearQuest 发送一个缺陷，WebInspect 会自动给会话信息添加一条注释 |
| XML Request<br>(XML 请求) | 显示嵌入在请求(在 Web 服务评估选择的 WSDL 对象时可用)中的 XML 架构 |
| XMLResponse<br>(XML 响应) | 显示嵌入在响应(在 Web 服务评估选择的 WSDL 对象时可用)中的 XML 架构 |
| Web Service Request<br>(Web 服务请求) | 显示 Web 服务架构和嵌入的请求值(在 Web 服务扫描中可用) |
| Web Service Response<br>(Web 服务响应) | 显示 Web 服务架构和嵌入的响应值(在 Web 服务扫描中可用) |

大多数选项在信息窗格顶部提供了搜索功能，方便用户找到指定的文本。要使用正则表达式进行搜索，单击 Find(查找)之前选择 Regex(正则表达式)选项。

注意：查看漏洞信息时，如果要打开链接，请在导航窗格中单击突出显示的会话。

### 3. 主机信息面板

当单击可折叠面板中列出的任意条目，WebInspect 显示网站(或主机)爬行或审计过程中发现该条目类型的所有实例。例如，如图 7-16 所示，用户选择主机信息面板中的 Cookie，WebInspect 将列出 Cookie 的所有会话。表 7-5 列出了主机信息面板选项。

**表 7-5　主机信息面板选项**

| 选　项 | 定　义 |
| --- | --- |
| E-mails(电子邮件) | WebInspect 列出在扫描过程中发现的所有电子邮件地址 |
| Forms(表单) | WebInspect 列出在扫描过程中发现的所有 HTML 表单 |
| P3P Info(P3P 信息) | 万维网联盟的个人隐私安全平台项目(P3P)用网站的标准格式来阐述常见的个人隐私，用户代理可自动检索且容易理解。P3P 用户代理允许用户在已知网站进行操作(在机器和人类的可读格式下)，在适当的时候基于这些操作进行自动化决策。因此，用户可以不必查看他们访问每个网站的隐私策略。P3P 官方网站声明了一个关于如何收集和使用这些信息的方案。支持 P3P 的 Web 浏览器可以通过这一方案与用户的存储偏好相比较来决定该怎么做。例如，用户可以设置浏览器偏好，以便他们浏览习惯的信息不被收集。当用户在随后访问网站的策略中规定一个 Cookie 被用于此目的时，浏览器会自动拒绝该 Cookie |

(续表)

| 选　项 | 定　义 |
|---|---|
| AJAX | AJAX 是异步 JavaScript 和 XMLHttpRequest 的缩写。如果选择此选项，WebInspect 可以显示包含一个 AJAX 引擎的所有页面，以及 AJAX 请求。AJAX 并不是技术本身，而是结合现有技术，包括 HTML 或 XHTML、CSS 样式表、JavaScript、文档对象模型、XML、XSLT 和 XMLHttpRequest 对象的组合。当这些技术结合在 AJAX 模型中，Web 应用程序都能够快速、增量式更新浏览器的用户界面，而不必重新加载整个页面 |
| AJAX | 而不是在会话开始加载一个网页，浏览器首先加载 AJAX 引擎，以便呈现用户界面，以及与服务器通信。每个用户操作，通常会产生一个 HTTP 请求，它需要有 AJAX 引擎的 JavaScript 来代替，而不需要与服务器(如简单的数据校验，内存中的数据编辑，甚至一些导航)通信，因此用户动作的任何响应都由引擎处理。如果引擎需要与服务器通信——提交数据进行处理，加载额外的界面代码，或者获取新数据——引擎通常使用 XML 让这些请求异步发送 |
| Certificates(证书) | 证书指出，一个特定的网站是安全的和真实的。它确保没有其他网站可以伪造原始安全网站的标识。当通过 Internet 发送个人信息时，用户应该检查该网站的证书，以确保它会保护个人身份信息。当从网站下载软件时，证书验证该软件是否来自一个已知的可靠来源。<br>证书与公钥的身份相关联。该证书只有拥有者知道相应的私有密钥，这使得拥有者可以做一个“数字签名”或使用相应的公钥加解密信息 |
| Comments(评论) | 通常情况下，开发人员在注释中留下的重要信息，可能会破坏网站的安全性 |
| Hiddens(隐藏) | WebInspect 分析所有形式，然后列出类型为“隐藏”(即控制那些没有渲染，但其值用表单提交)的所有控件。开发人员通常使用的参数隐藏控件可被攻击者编辑后重新提交 |
| Cookies | Cookie 包含存储在客户端上以供将来交互的信息(如用户首选项或配置信息)。Cookie 存在两种基本形式：作为单独文件或者是一个连续的文件记录。通常是多个被安装在不同位置的浏览器产生信息的集合。许多情况下，“被遗忘的”Cookies 包含用户不希望别人看到的信息。 |
| Scripts(脚本) | 脚本是基于浏览器中的输入请求，在服务器上运行和处理的程序。WebInspect 可跟踪每个脚本，运行并列出信息选项卡上的所有脚本 |
| Broken Links(断开的链接) | WebInspect 可发现并记录网站上的所有处于非工作状态的超链接 |
| Offsite Links (异地链接) | WebInspect 可发现并记录所有到其他网站的超链接 |
| Parameters(参数) | WebInspect 可发现并记录在扫描过程中遇到的所有参数。参数是在 HTTP 请求中提交的 URL 的一部分查询字符串或者使用 POST 方法提交的数据 |

双击列表中的条目，WebInspect 可在导航窗格中突出显示包含该条目的会话。用户可通过突出显示文本，从快捷菜单中选择 Copy(复制)将条目(如电子邮件地址)复制到剪贴板。

注意：在进行 Web 服务扫描时，主机信息面板不显示。

大多数情况下，用户可在信息窗格顶部使用搜索功能找到指定的文本。要使用“正则表达式”进行搜索，单击 Find(查找)之前选择 Regex(正则表达式)选项。

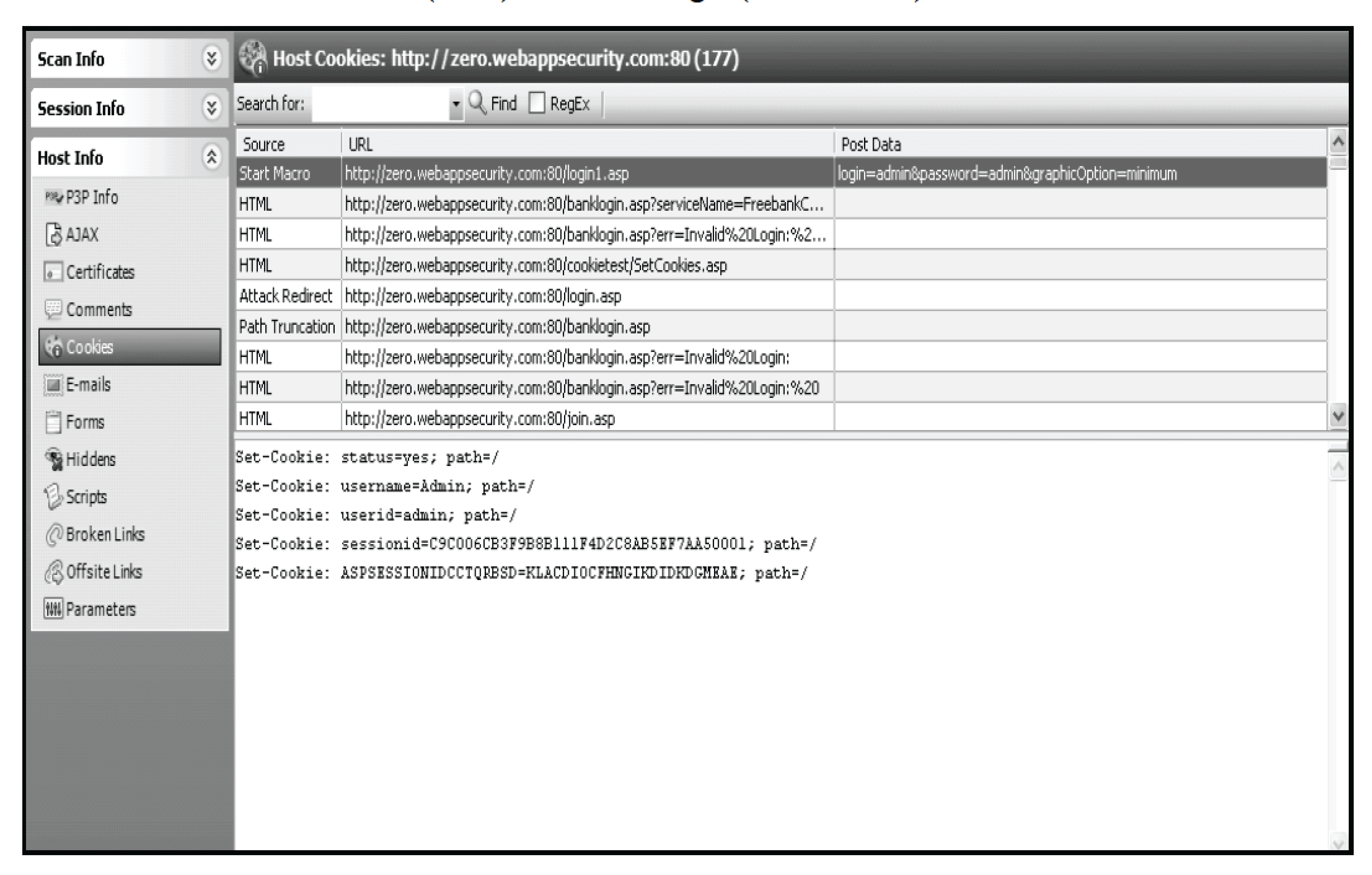



图 7-16　主机信息面板中的 Cookie

## 7.2.4　摘要窗格

在进行或查看扫描时，使用位于窗口底部的水平摘要窗格，可显示发现的全部漏洞。从而可以快速访问漏洞信息和查看 WebInspect 记录信息。此窗格有六个选项卡，如图 7-17 所示。

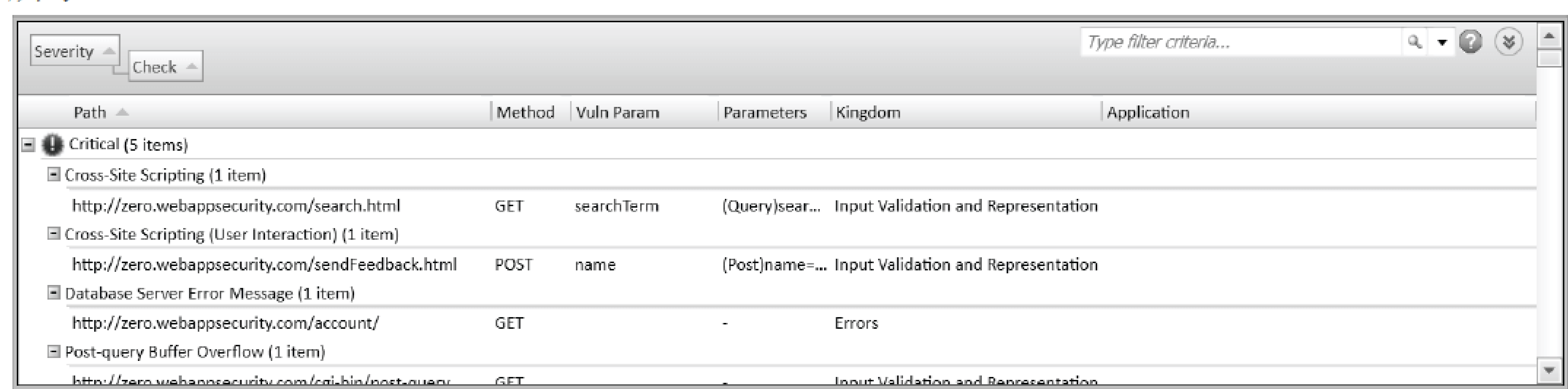



图 7-17　摘要窗格的 6 个选项卡

- 漏洞(Vulnerabilities)
- 未发现(Not Found)
- 信息(Information)
- 最佳实践(Best Practices)

- 扫描日志(Scan Log)
- 服务器信息(Server Information)

**注意**　用户可在除了扫描日志的所有选项卡上进行分组和结果过滤。

### 1. 漏洞选项卡

该漏洞选项卡列出了审计期间在网站上发现的所有漏洞。

如果要选择显示的信息，从快捷菜单中右键单击列标题栏选择 Columns(列)，如图 7-18 所示。

可用列如下：

- 严重性(Severity)：该漏洞的相对评价，从低到高。请参阅下面的相关图标。
- 检查(Check)：WebInspect 可探测特定的漏洞，例如跨网站脚本，未加密的日志通知等。
- 检查 ID(Check ID)：WebInspect 探头用于检查特定漏洞存在的标识号。例如，检查编号 742 测试数据库服务器的错误消息。
- 路径(Path)：分层路径资源。
- 方法(Method)：用于攻击的 HTTP 方法。
- 堆栈(Stack)：从 HP Fortify Software 安全中心获得的堆栈跟踪信息。此列仅当 SecurityScope 在扫描过程中启用。
- 漏洞参数(Vuln Param)：有漏洞的参数名称。
- 参数(Parameters)：分配的参数和值名称。
- 发布状态(Published Status)：存在于软件安全中心以前发布的状态。
- 挂起状态(Pending Status:)：该扫描还没有发布的状态(由 WebInspect 手动或自动分配)。
- 手动(Manual)：如果该漏洞被手动创建，显示一个复选标记。
- 重复(Duplicates)：通过 SecurityScope 检测到的漏洞可以追溯到同一个源头。
- 地点(Location)：路径和揭示漏洞的数据。
- CWE ID：与漏洞相关的共同脆弱性的枚举标识符(次)。
- 领域(Kingdom)：在这个漏洞被归类时，使用 Fortify Software 安全研究小组开发的软件安全性错误的分类类别。

  单击超链接导航到http://www.hpenterprisesecurity.com/vulncat/en/vulncat/index.html。

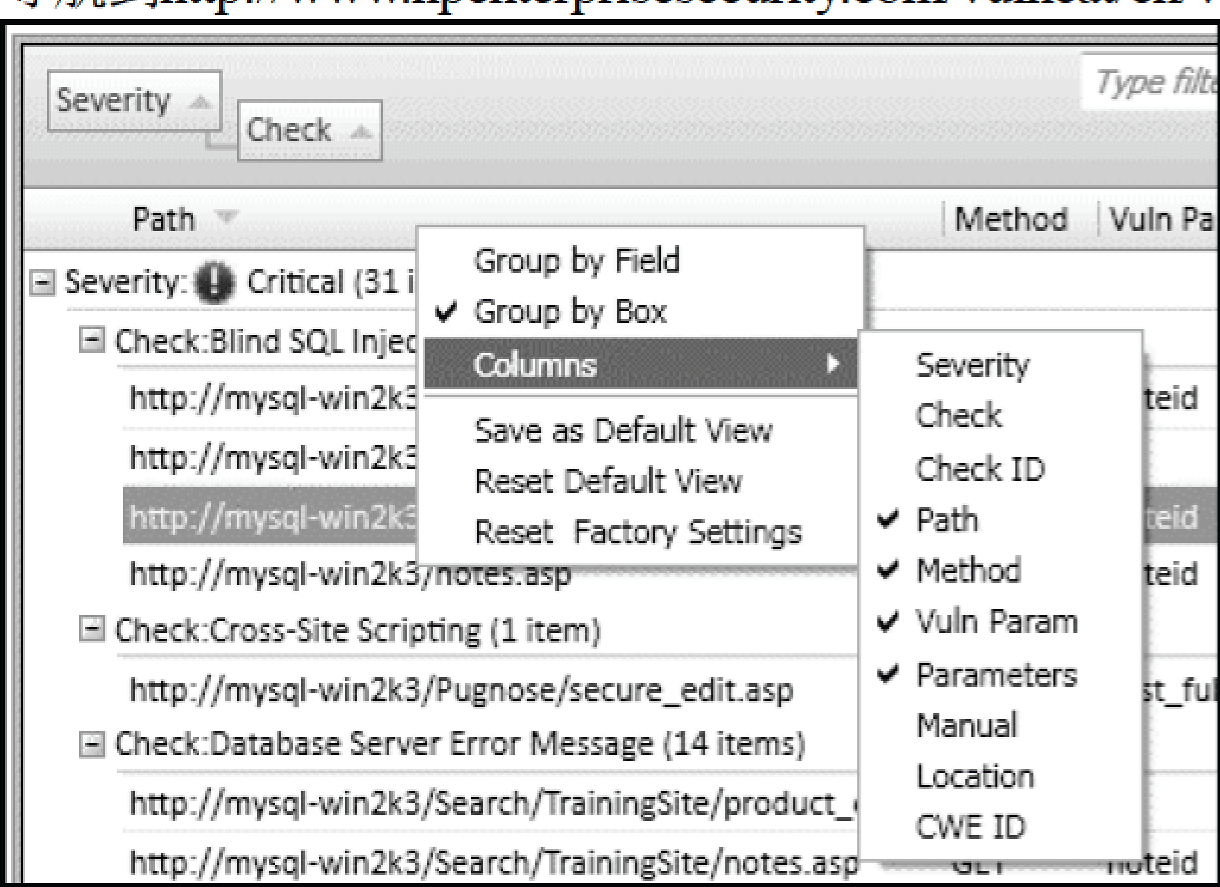



图 7-18　漏洞选项卡右键快捷菜单

- 重复性(Reproducible)：值可以被复制。此列仅适用于网站的重复试验(验证漏洞)。

漏洞的严重程度由以下图标表示，如图 7-19 所示。

| Critical | High | Medium | Low |
| --- | --- | --- | --- |

图 7-19　漏洞的严重程度

如果单击列表中的一个条目，信息窗格中的相关信息以及会话会突出显示出来。

选中会话，也可以通过从会话信息面板中选择一个选项，查看相关信息。

对于发送和查询参数，单击参数列中的条目，将显示该参数的一个更可读的概要。

用鼠标右键单击列表中的条目，会显示下面列表中所描述命令的快捷菜单。

使用快捷菜单，右键单击漏洞选项卡上的漏洞描述，用户可以：

- 复制网址(Copy URL)：复制网址到 Windows 剪贴板。
- 复制选择条目(Copy Selected Item(s))：复制选定的条目到 Windows 剪贴板中的文本。
- 复制所有条目(Copy All Items)：复制所有的条目到 Windows 剪贴板中的文本。
- 导出(Export)：创建一个逗号分隔值(CSV)文件，其中包含所有或选定条目，并在 Microsoft Excel 中显示。
- 在浏览器中查看(View in Browser)：在浏览器中呈现 HTTP 响应。
- 当前值的过滤器(Filter by Current Value)：限制那些满足选择标准显示的漏洞。例如，如果在 Post 方法列单击鼠标右键，然后选择 Filter by Current Value，列表中只显示那些通过使用 POST 方法发送的被发现的 HTTP 请求漏洞。
- 更改发布状态(Modify Publish Status)：发布到 HP Fortify Software 安全中心之前可以更改漏洞/问题的状态。只有当连接到 WebInspect Enterprise，该命令才可用。
- 更改严重性(Change Severity)：允许更改严重性级别。
- 编辑漏洞(Edit Vulnerability)：允许添加、删除或编辑与选定会话相关的漏洞。
- 审阅漏洞(Review Vulnerability)：重新测试漏洞会话，将其标记为误报或忽视，或将其发送到 HP 质量中心或 IBM Rational ClearQuest。
- 标记为(Mark As)：允许作为误报标记漏洞或忽略。要查看误报，在扫描信息面板中选择误报。查看忽略的漏洞，在扫描信息面板中选择对象，单击删除条目中相关联的数字，当恢复已删除条目窗口出现，从下拉列表中选择漏洞。
- 发送到(Send To)：转换该漏洞的缺陷，并将其添加到 HP 质量中心或 IBM Rational ClearQuest 数据库。
- 删除位置(Remove Location)：从导航窗格中删除会话。注意：用户可恢复删除的会话。
- 爬行(Crawl)：爬行选定的 URL。
- 工具(Tools)：介绍可用工具的子菜单。
- 附件(Attachments)：允许创建与选定会话相关的说明，标志会话跟进，添加一个漏洞说明，或添加一个漏洞快照。

如果右键单击一个组标题，通过一个快捷菜单可以进行如下操作：

- 折叠/展开所有组(Collapse/Expand All Groups)
- 折叠/展开组(Collapse/Expand Group)
- 复制选定的条目(Copy Selected Item(s))
- 复制所有条目(Copy All Items)
- 更改严重性(Change Severity)
- 标记为(Mark as)
- 发送到(Send to)
- 删除位置(Remove Location)

WebInspect 可从导航窗格(在这两个网站和序列视图)和摘要窗格中的 Vulnerability 选项卡中删除条目。它也忽略了这些条目的任何报告。删除的条目数量会显示在仪表板上(在扫描类别下)。用户可利用下面的过程恢复删除的会话和忽略的漏洞。

1) 单击出现在删除条目标题附近突出显示的数目。恢复已删除条目窗口显示删除会话或删除漏洞列表。

2) 单击下拉列表在忽略漏洞和删除会话之间进行切换。

3) 选中要恢复一个或多个条目的复选框。

4) 要查看时选择条目的详细信息，选择 Show details when selected(显示详细信息)。

5) 当系统提示用户确认选择时，单击 Recover(恢复)，然后单击 Yes。

恢复漏洞重新出现在导航窗格中网站和序列视图上(连同他们的父会话)，也重新出现在摘要窗格中 Vulnerabilities(漏洞)选项卡上。在导航窗格中恢复会话也随着子会话和他们的漏洞再次出现。

进行扫描后，报告会发现漏洞，开发者可以纠正他们的代码并更新网站。然后，用户可打开原始扫描，选择历史漏洞会话(once-vulnerable session，也可称为修复)，然后从快捷菜单中选择 Review Vulnerability(审阅漏洞)。假设该网站的基本架构并没有改变，则不必重新扫描整个网站(某些情况下，可能需要几个小时甚至几天)，用户验证的威胁也不再存在。

使用此功能只是为了再次检查报告的漏洞，即使扫描仍在运行中也可以进行检查。

1) 从导航窗格(或右键单击摘要窗格的漏洞选项卡上的 URL)右键单击一个漏洞会话。

2) 从快捷菜单中选择 Review Vulnerability(审阅漏洞)。打开审阅漏洞窗口。

3) 如果要通过 Web 代理服务器来访问网站，单击 Options(选项)，选择 Launch and Direct Traffic through Web Proxy(Web 代理启动和直接流量)。

4) 如果有多个安全漏洞与选定会话相关联，从 Vulnerabilities to Test(测试列表)列表选择一个。

5) 使用右侧客户区的选项卡以显示有关原始(original)会话信息(如在 URL 列下的低窗格中选定)：

浏览器：服务器的响应，如在浏览器中呈现。

请求：原始的 HTTP 请求消息。

响应：原始的 HTTP 响应消息。

漏洞：漏洞说明，影响和建议如何解决此漏洞。

6) 要重新测试选定漏洞的会话，单击 Retest(重新测试)。

重新测试的结果显示在状态栏和 Response Match Status(响应匹配状态)列下方的窗格中。

状态报告为“Vulnerability<VulnerabilityName>Detected”或“Vulnerability <Vulnerability Name> Not Detected。”

所报告结果的可靠性由响应匹配状态决定，可具有以下值：

匹配(Match)：资源并没有显著改变；WebInspect 能通过使用原来路径相同的扫描来访问会话。

无结论(Inconclusive)：基于 HTTP 响应，该资源用可能(或不可能)的方式来标识漏洞重新检测过程中有或没有检测到改变。

不同(Different)：HTTP 响应与从原始扫描过程中接收到的响应是完全不同的，提示资源发生重大变化。

7) 如果用户认为 WebInspect 错误地判断了漏洞存在，可通过单击 Mark as False Positive (标记为误报)标出漏洞。

8) 要忽略漏洞，单击 Mark as(标记为)，然后从下拉列表中选择 Ignored(忽略)。

9) 要转换一个或多个漏洞的缺陷，并将其添加到惠普质量中心或 IBM Rational ClearQuest 数据库，单击 Send To(发送到)。

**注意**　如果从漏洞比较窗口中访问漏洞审阅窗口，Mark As(标记为)和 Send To(发送到)按钮未启用。

### 2. 未发现选项卡

只有当 WebInspect 连接到 WebInspectEnterprise 或评估管理平台(AMP)，此标签才有密切关系，目的是用于同步和上传漏洞数据到 HP Fortify Software 安全中心。它列出了一个特定的条目版本，这个版本是以前扫描检测到的漏洞，而不是由当前扫描检测到的。这些漏洞未被记录在仪表板的计数中，也未出现在导航窗格中的网站或序列视图中。

在快捷菜单中的选项，分组和筛选功能与 Vulnerabilities(漏洞)选项卡中的描述相同。

### 3. 信息选项卡

信息选项卡列出了 WebInspect 评估或爬行过程中发现的信息。这些不被视为安全漏洞，而是在网站、某些应用程序或 Web 服务器识别的关注点。单击列出的 URL，该程序突出显示了导航窗格中的相关条目。

在快捷菜单中的选项，分组和筛选功能与 Vulnerabilities(漏洞)选项卡中的描述相同。

### 4. 最佳实践选项卡

WebInspect 检测出的最佳实践选项卡列表问题涉及被 Web 开发者共同接受的最佳实践。这里列出的条目不是漏洞，而是整体网站质量和网站开发的安全实践(或缺乏)指标。

在快捷菜单中的选项，分组和筛选功能与 Vulnerabilities(漏洞)选项卡中的描述相同。

### 5. 扫描日志选项卡

使用 Scan Log(扫描日志)选项卡以查看有关 WebInspect 评估行为的信息。比如，应用于针对网站存在的特定审计方法的时间将在这里列出。用户可在应用程序设置窗口中使用

Logging(日志)记录选项选择日志记录级别(调试、信息、警告、错误或致命)，如图 7-20 所示。

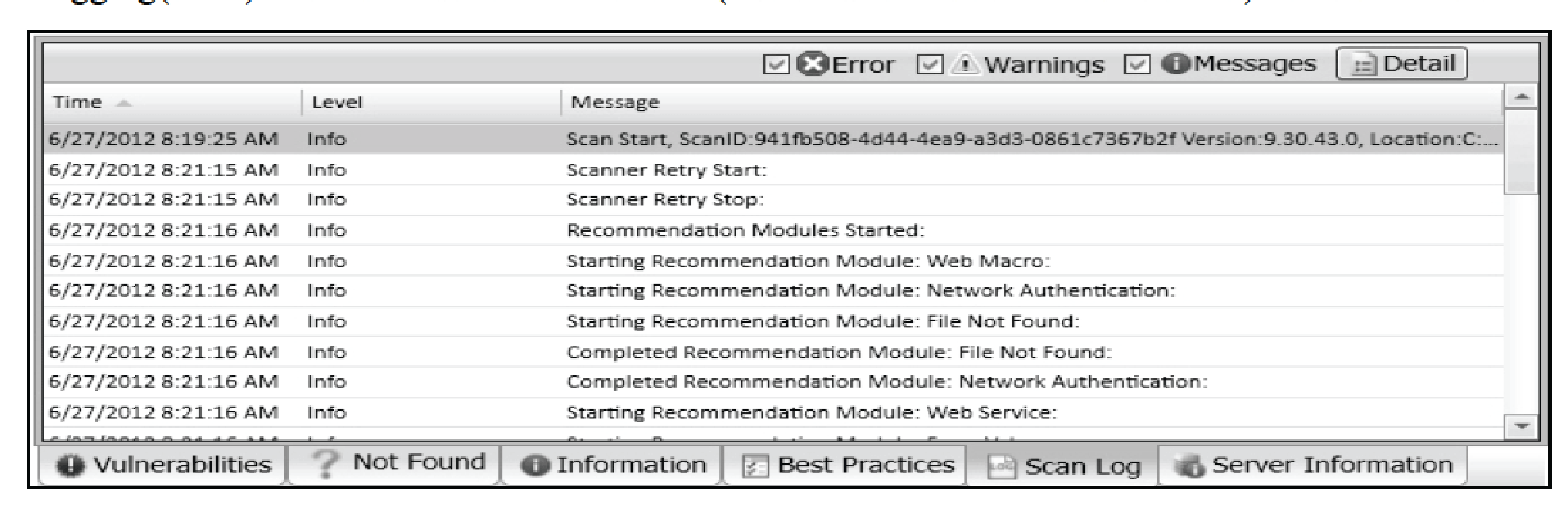



图 7-20　扫描日志选项卡

### 6. 服务器信息选项卡

这个选项卡列出关注的相关服务器条目。这里仅列出一个条目或事件的发生。

#### 1) 使用过滤器

用户可查看符合以下指定两种方法之一的条目的子集：

- 在窗格的右上角使用组合框输入筛选条件。

**注意**　单击过滤标准框，然后按 Ctrl +空格键可查看所有可用的过滤器条件的弹出列表，然后输入该标准的值。

- 右键单击任一列的值，并从快捷菜单中选择 Filter by Current Value(当前值过滤器)。这种过滤能力适用于所有摘要窗格中的选项卡，除了 Scan Log(扫描日志)。

在下面的例子中，第一个画面显示了漏洞选项卡上未过滤的条目。第二个画面显示过滤条件中的“Method:Get”。

无过滤条件，如图 7-21 所示。

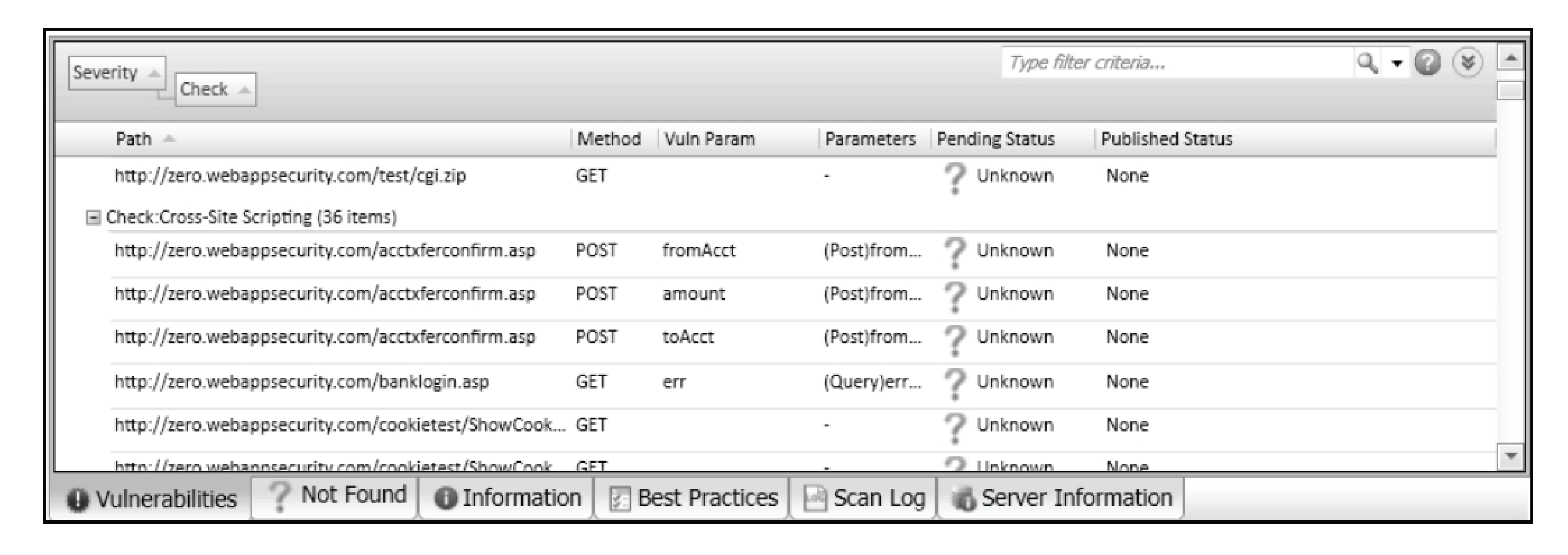



图 7-21　漏洞选项卡上无过滤条件

过滤条件：Get，如图 7-22 所示。

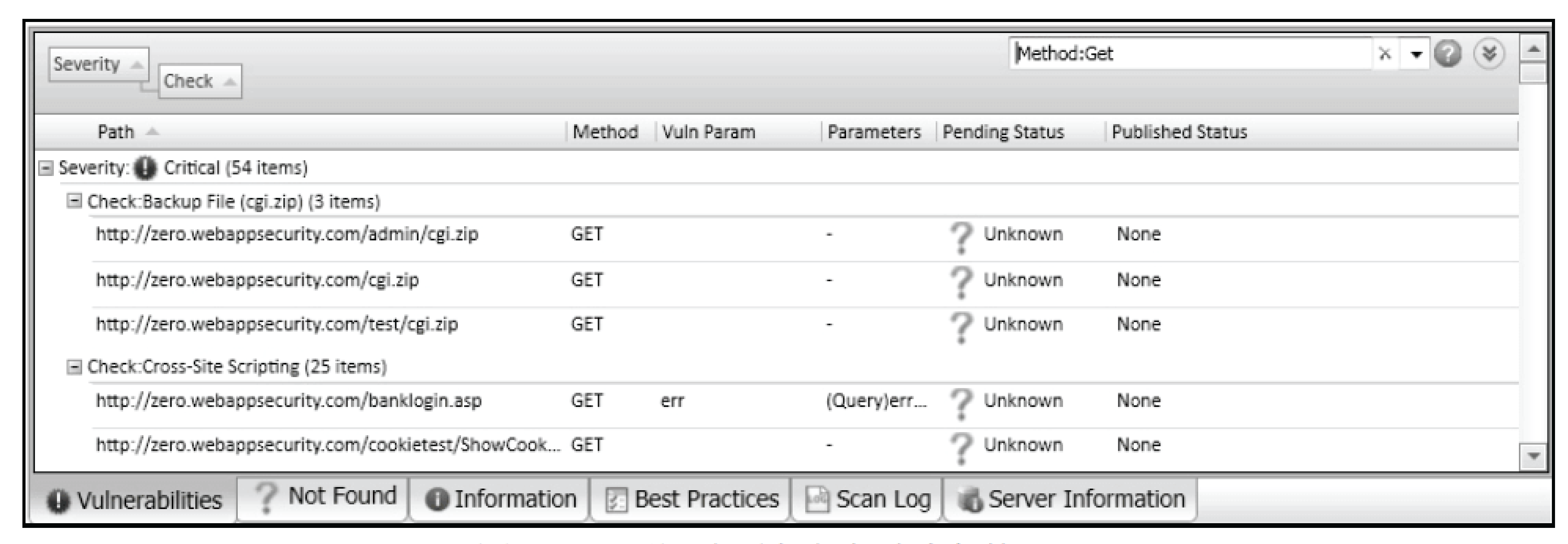



图 7-22　漏洞选项卡上有过滤条件：Get

请注意，过滤条件(方法：Get)出现在组合框，其中也包含了红色的 X。单击它来去除过滤和返回到原来列表。

在过滤组合框中键入条件时指定多个过滤条件，插入过滤条件之间用逗号隔开(如参数：noteid，方法：GET)。

筛选条件：可输入以下标识：

- 检查(check)：检查名称；
- Cookienamerp：HTTP 响应中的 Cookie 名称；
- cookienamerq：HTTP 请求中的 Cookie 名称；
- cookievaluerp：HTTP 响应中的 Cookie 值；
- cookievaluerq：HTTP 请求中的 Cookie 值；
- 重复(duplicates)：通过 SecurityScope 重复检测；
- filerq：HTTP 请求文件名和扩展名；
- headernamerp：HTTP 响应头名；
- headernamerq：HTTP 请求报头名；
- headervaluerp：HTTP 响应头值；
- headervaluerq：HTTP 请求头值；
- 位置(location)：路径加上参数识别资源；
- 手动(manual)：一个位置手动添加(语法手册：True，或手动：False)；
- 方法(method)：HTTP 方法(GET，POST)；
- methodrq：在 HTTP 请求中指定的方法；
- 参数(parameters)：在 HTTP 请求中指定的参数；
- 路径(path)：路径识别资源(不带参数)；
- Rawrp：原始 HTTP 响应；
- Rawrq：原始 HTTP 请求；
- sessiondataid：会话数据的标识符；
- 严重性(severity)：分配给一个漏洞(关键，高，中，低)的严重性；
- 栈(stack)：通过 SecurityScope 返回堆栈跟踪(语法是堆栈：True 或堆栈：False)；
- StatusCode：HTTP 状态代码；
- Typerq：请求的类型：query、post 或 SOAP；
- Vparam：该漏洞参数。

### 2) 使用组

用户可将条目转换为基于列标题的类别。要做到这一点，在窗格的顶部只需拖动标题并拖放到分组区域。图 7-23 中的漏洞是先通过风险、然后按照名称字母顺序进行的分组。

如果右键单击列标题，WebInspect 可以显示下列快捷菜单：

- 领域分组(Group by Field)：根据所选择的领域将漏洞分组。
- 框分组(Group by Box)：显示“分组依据”区域中，用户可以通过列标题进行分组。
- 列(Columns)：允许选择显示哪一列。
- 另存为默认视图(Save as Default View)：保存当前的分组模式作为默认视图并全部扫描。

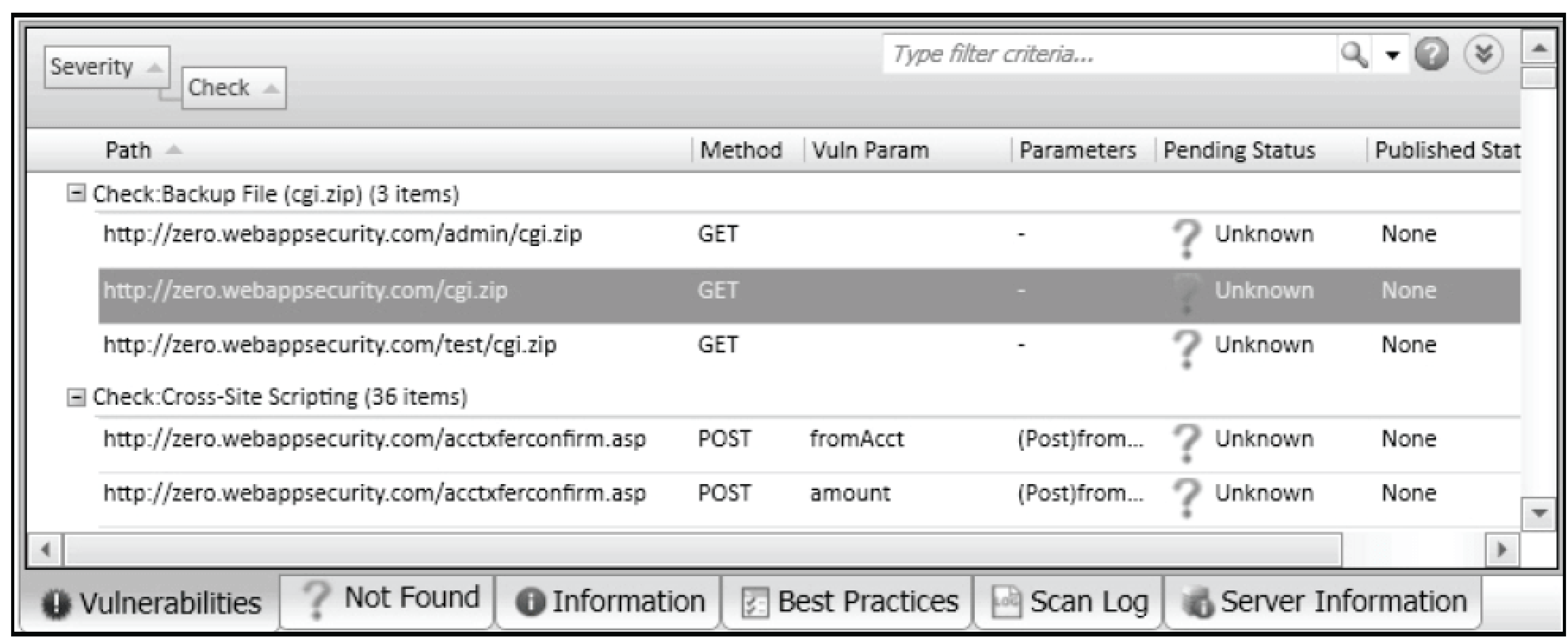



图 7-23　选中漏洞

- 重置默认视图(Reset Default View)：恢复分组范式到初始化的默认视图。
- 恢复出厂设置(Reset Factory Settings)：恢复分组范式到原始视图模式(严重性>检查)。

## 7.2.5　WebInspect 工具栏

WebInspect 窗口包含两个工具栏：扫描和标准。用户可通过选择 View→Toolbars(视图→工具栏)显示或隐藏任何工具栏。

表 7-6 列出了 WebInspect 扫描工具栏上的可用按钮。

表 7-6　扫描工具栏按钮

| 按　钮 | 功　能 |
|---|---|
| Start / Resume | 用户可以暂停扫描，然后继续扫描。此外，完成扫描可能包含未发送(因为超时或其他错误)的会话；如果单击 Start(开始)，WebInspect 将试图重新发送这些会话 |
| Pause | 中断正在进行的扫描。用户可通过单击开始/恢复图标继续扫描 |
| Skip | 当进行连续爬行和审计时，用户可跳过引擎正在运转的处理部分(如果选择 Test each engine type per session(测试每个会话的每个引擎类型))或跳过扫描处理的会话(如果选择 Test each session per engine type(测试每台引擎类型的每个会话)) |

(续表)

| 按　钮 | 功　能 |
|---|---|
| Audit | 如果进行只爬行扫描或步模式扫描，可在扫描后单击此图标进行审计 |
| Rescan | 仅当选择扫描选项卡时显示此按钮。如果从下拉菜单中选择 Scan Again(再次扫描)，它会启动上次使用扫描时预先设置的扫描向导。如果选择 Verify Vulnerabilities(验证漏洞)，则只对原始扫描到有漏洞的目标网站进行检测 |
| Compare | 仅当选择扫描选项卡时显示此按钮。可通过比较同一目标的两个不同扫描发现漏洞 |
| Synchronize | 此按钮仅连接到 WebInspect Enterprise 后出现。它允许指定软件安全中心(SSC)和版本。WebInspect 可从 SSC 再次下载漏洞列表，比较下载的漏洞和当前扫描的漏洞，为当前扫描到的漏洞分配一个合适的状态(新建、现有、重新或者未找到) |
| Publish | 此按钮仅连接到 WebInspect Enterprise 后出现，当用户已同步到 WebInspect 软件安全中心后并启用。它通过 WebInspect 将项目版本数据上传到企业软件安全中心 |
| Run in AMP | 此按钮仅在 WebInspect 连接到评估管理平台(AMP)时出现在打开的扫描选项卡中。它允许将扫描设置发送到 AMP，AMP 创建一个扫描请求，并发送到扫描队列中的下一个可用检测部件 |
| Run in WebInspect Enterprise | 此按钮仅出现在 WebInspect 连接到 WebInspect Enterprise 且扫描具有焦点时。它允许将扫描设置发送到 WebInspect Enterprise，WebInspect Enterprise 创建一个扫描请求，并发送到扫描队列中的下一个可用检测部件 |

表 7-7 列出了 WebInspect 标准工具栏上的可用按钮。

表 7-7　标准工具栏按钮

| 按　钮 | 功　能 |
|---|---|
| New | 允许启动向导扫描、网站扫描、Web 服务扫描或企业扫描 |
| Open | 允许打开一个扫描或报告 |
| Compliance Manager | 打开合规管理器，允许创建一个合规政策 |
| Policy Manager | 打开策略管理器，允许修改 WebInspect 的扫描策略，并创建自定义检查，以检查特定漏洞 |
| Report | 如果单击 start 页面具有焦点时，WebInspect 可以提示扫描，然后允许选择报告。如果单击扫描选项卡具有焦点时，会提示选择要打开的扫描报告 |
| Schedule | 安排发生在特定时间和日期的扫描 |

(续表)

| 按　钮 | 功　能 |
| --- | --- |
| SmartUpdate | 联系 HP 中央数据库，以确定系统是否有可用的更新，如果存在更新，提示用户安装 |
| AMP WebConsole | 启动 AMP 的 Web 控制台应用程序。只有当用户连接到 AMP 时显示此按钮 |
| WebInspect Enterprise WebConsole | 启动 WebInspect EnterpriseWeb 控制台应用程序。只有当用户连接到 WebInspect Enterprise 时显示此按钮 |

表 7-8 说明了管理扫描工具栏上可用的按钮。

表 7-8　管理扫描工具栏按钮

| 按　钮 | 功　能 |
| --- | --- |
| Open | 要打开扫描，选择一个或多个扫描，然后单击 Open(或直接双击列表中的条目)。WebInspect 可以加载扫描数据并显示每次扫描的单独选项卡 |
| Rescan ▾ | 要启动预先设置的最近用于选定扫描的扫描向导，单击 Rescan→ Scan Again(重新扫描→再次扫描)。仅重新扫描那些以前扫描过程中发现漏洞的会话，选择一个扫描，并单击 Rescan→ Retest Vulnerabilities(重新扫描→重新测试漏洞) |
| Rename | 要重新命名选定的扫描，单击 Rename(重命名) |
| Delete | 要删除选定的扫描，单击 Delete(删除) |
| Import | 要导入扫描，单击 Import(导入) |
| Export ▾ | 要导出扫描，导出扫描的详细信息，或导出扫描到软件安全中心，单击 Export(导出)的下拉箭头 |
| Compare | 为比较扫描，选择两个扫描(使用 Ctrl +单击)，然后单击 Compare(比较) |
| Connections | 默认情况下，WebInspect 列出保存在本地 SQL Server Express Edition 和配置 SQL Server 标准版的所有扫描。要选择一个或两个数据库，或指定一个 SQL Server 连接，单击 Connections(连接) |
| Refresh | 必要时，单击 Refresh(刷新)以更新显示 |
| Columns | 单击 Columns(列)选择要显示哪些列。也可以重新排列各列顺序，使用上移下移按钮，或者在管理扫描列表中，可以简单地拖放列标题 |

## 7.2.6　WebInspect 菜单栏

菜单栏包含以下菜单：

- 文件(File)
- 扫描(Scan)
- 编辑(Edit)
- 企业服务器(Enterprise Server)

- 视图(View)
- 报告(Reports)
- 工具(Tools)
- 帮助(Help)

#### 1) 文件菜单

下面的命令在文件菜单中可用。

新建(New)：允许用户选择向导扫描、基础扫描、Web 服务扫描或企业扫描，从而启动相应的向导，引导用户完成扫描的过程。

打开(Open)：允许打开任何一个扫描或报告。

计划(Schedule)：打开管理扫描调度窗口，允许添加、编辑或删除计划扫描。

导入扫描(Import Scan)：允许导入扫描文件。

导出(Export)：让用户导出扫描文件或扫描详细信息文件。

关闭选项卡(Close Tab)：当多个选项卡打开时，关闭当前标签。

退出(Exit)：关闭 WebInspect 程序。

#### 2) 编辑菜单

下面的命令在编辑菜单中可用。

默认扫描设置(Default Scan Settings)：显示默认的设置窗口，允许用户选择或修改用于扫描的选项。

当前扫描设置(Current Scan Settings)：显示一个设置窗口，允许用户选择或修改当前扫描选项。

管理设置(Manage Settings)：打开一个窗口，允许用户添加、编辑或删除设置文件。

应用程序设置(Application Settings)：显示应用程序设置窗口，允许用户选择或修改控制 WebInspect 应用程序操作选项。

复制网址(Copy URL)：将选定的 URL 复制到 Windows 剪贴板。

复印扫描日志(Copy Scan Log)：将日志(所选选项卡上的扫描)复制到 Windows 剪贴板。

#### 3) 视图菜单

下面的命令在 View 菜单中可用。

自动换行(Word Wrap)：当查看 HTTP 请求和响应时，在显示区域的右侧边距插入回车。只有当选中扫描选项卡时，该命令才可用。

工具栏(Toolbars)：允许用户选择要显示的工具栏。

#### 4)工具菜单

工具菜单命令用来启动被 WebInspect 封装的工具应用程序。

#### 5) 扫描菜单

只有当选中扫描选项卡具有焦点，扫描菜单才是可见的。它包含下面的命令。

开始/恢复(Start/Resume)：用户暂停程序后，开始或恢复扫描。

暂停(Pause)：暂停爬行或审计程序。单击 Resume(恢复)继续扫描。

跳过(Skip)：如果审计正在进行中，跳到下一个审计方法。如果在这个过程中正在进行爬行，跳过这个审计。

审计(Audit)：评估爬行网站的漏洞。完成爬行或退出步模式之后使用该命令。

重新扫描(Rescan)：此命令将启动上次所选扫描预先设定的扫描向导。

#### 6) 企业服务器菜单

Enterprise Server 菜单包含以下命令。除了 Connect to AMP 的所有命令，仅当 WebInspect 连接到一个企业服务器时，Connect to WebInspect Enterprise 以及 About Enterprise Servers 是可用的。

连接到 AMP/断开(Connect to AMP/Disconnect)：建立或断开评估管理平台(AMP)服务器的连接。

连接到 WebInspect Enterprise/断开(Connect to WebInspect Enterprise/Disconnect)：建立或断开 WebInspect Enterprise(WIE)服务器的连接。

上传扫描(Upload Scan)：用于将数据传送到 AMP 服务器进行扫描，供用户选择。在“自动上传扫描”未设置时，这是使用频率最高的操作。

下载扫描(Download Scan)：允许用户选择从 AMP 服务器复制到硬盘驱动器的扫描。

发布扫描(Publish Scan)：显示一个对话框，允许用户查看漏洞，并将它们传送到企业服务器，再将它们依次传送到 HP Fortify Software 的安全中心服务器。

传送设置(Transfer Settings)：允许选择一个 WebInspect 设置文件，并将其转移到企业服务器，这将创建一个基于这些设置的扫描模板。还可选择一个企业级服务器扫描模板，并将其转移到 WebInspect，这将创建一个基于模板的设置文件。

Web 控制台(WebConsole)：启动 AMP 或 WebInspect Enterprise Web 控制台应用程序。只有当连接到企业服务器该按钮会出现。

关于企业服务器(About Enterprise Server)：显示相关评估管理平台和 WebInspect Enterprise 信息。

注意，WebInspect 在任何时候都可以安装一个独立的许可证连接到企业服务器，只要用户是 AMP 或 WebInspect Enterprise 的角色成员。

#### 7) 报告菜单

报告菜单包含以下命令。

生成报告(Generate Report)：启动报告生成。

管理报告(Manage Reports)：显示标准和自定义报告类型的列表。用户可以重命名、删除或导出自定义设计报告，也可以导入报告定义文件。

报告设计器(Report Designer)：启动报告设计器，允许用户定义一个自定义报告。

#### 8) 帮助菜单

下面的命令在帮助菜单中可用。

WebInspect 帮助(WebInspect Help)：打开帮助文件。

首页(Index)：打开帮助文件，在左窗格中显示索引。

搜索(Search)：打开帮助文件，在左窗格中显示搜索选项。

支持(Support)：如果支持渠道被启用，显示一个视窗，允许用户将增强请求提交给 HP。

教程(Tutorials)：允许下载教程和其他 WebInspect 文档。

关于 WebInspect(About WebInspect)：有关 WebInspect 应用程序显示的信息，包括许可证信息、允许主机和属性。

### 7.2.7 HP 应用安全中心

HP 应用安全中心的应用程序，在任务栏通知区域用图标表示，如图 7-24 所示，提供一个上下文菜单，允许用户：

- 开始/停止检测部件服务。
- 开始/停止调度服务。
- 配置 WebInspect 作为评估管理平台(AMP)检测部件。

当某些事件发生时，弹出消息也会随之出现。

使用这项功能，用户可将 WebInspect 作为一个独立的扫描仪进行安装，然后连接到 AMP 或 WebInspectEnterprise。

### 7.2.8 检查结果：向导扫描或基础扫描

只要运行一个向导扫描或基础扫描，WebInspect 就会开始评估 Web 应用程序并在导航窗格中显示每个会话(使用本网站或序列视图)的图标。它还报告摘要窗格中漏洞选项卡和信息选项卡上可能存在的漏洞。

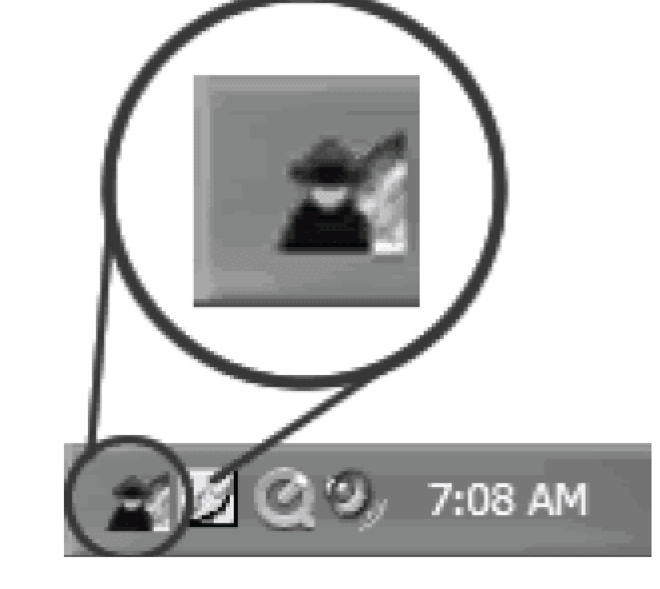


图 7-24 HP 安全中心的应用程序

如果单击摘要窗格中列出的 URL，在导航窗格中该程序突出显示相关会话，并在信息窗格中显示相关信息。

有时，所检测到有漏洞的会话攻击不会在攻击信息中列出。也就是说，如果用户在导航窗格中选择一个有漏洞的会话，然后在 Session Info(会话信息)面板单击 Attack Info(攻击信息)，攻击信息不会显示在信息窗格中。这是因为攻击信息通常与创造攻击的会话相关联，而不是与其被检测到的会话相关联。发生这种情况时，选择上一级会话，然后单击 Attack Info(攻击信息)。

如果在漏洞选项卡的漏洞描述上单击鼠标右键，用户可以通过快捷菜单：

- 复制网址(Copy URL)：复制网址到 Windows 剪贴板。
- 复制选择条目(Copy Selected Item(s))：复制选定的条目到 Windows 剪贴板中的文本。
- 复制所有条目(Copy All Items)：复制所有的条目到 Windows 剪贴板中的文本。
- 导出(Export)：创建一个逗号分隔值(CSV)文件。
- 在浏览器中查看(View in Browser)：在浏览器中查看 HTTP 响应。
- 当前值的过滤器(Filter by Current Value)：限定满足选择标准的漏洞显示。例如，如果在 POST 的方法列单击鼠标右键，然后选择 Filter by Current Value，列表中只显示那些被发现通过使用 POST 方法发送的 HTTP 请求的漏洞。

**注意** 过滤条件显示在摘要窗格右上角的组合框中。或者，也可以手动输入或选择使用组合框限制一个过滤条件。

- 更改严重性(Change Severity)：允许更改的严重性级别。
- 编辑漏洞(Edit Vulnerability)：显示编辑对话框中的漏洞，修改各种漏洞特征。
- 审阅漏洞(Review Vulnerability)：重新测试漏洞会话，将其标记为误报，或将其发送到 HP 质量中心或 IBM Rational ClearQuest。
- 标记为(Mark as)：允许标记漏洞作为误报或忽略。要查看误报，在扫描信息面板中选择误报。这两种情况下，该漏洞可从列表中删除。用户可在扫描信息面板选择误报来查看所有误报的列表。用户也可以通过在扫描信息面板中选择仪表板查看误报和忽略的漏洞列表，然后在统计列单击已删除条目的超链接数量。注意：可恢复“误报”和“忽略”的漏洞。
- 发送到(Send to)：转换该漏洞的缺陷，并将其添加到 HP 质量中心或 IBM Rational ClearQuest 数据库。
- 删除位置(Remove Location)：从导航窗格(包括网站和序列视图)删除选定会话，并删除任何相关的漏洞。注意：可恢复删除位置(会话)及其相关的漏洞。
- 爬行(Crawl)：爬行选定的 URL。
- 工具(Tools)：介绍可用工具的子菜单。
- 附件(Attachments)：允许创建与选定会话相关的说明，标志会话的跟进，添加一个漏洞说明，或添加一个漏洞快照。

如果右键单击组标题，通过一个快捷菜单可以：

- 折叠/展开所有组(Collapse/Expand All Groups)
- 折叠/展开组(Collapse/Expand Group)
- 复制选定的条目(Copy Selected Item(s))
- 复制所有条目(Copy All Items)
- 导出(Export)
- 更改严重性(Change Severity)
- 标记为(Mark as)
- 发送至(Send to)
- 删除位置(Remove Location)

### 1. 会话

会话是一个匹配的集合，包括通过 WebInspect 发送 HTTP 请求来测试漏洞，以及从服务器接收的 HTTP 响应。爬行或扫描过程中采用的每个会话被列在导航窗格中。

在信息窗格中单击导航窗格中的会话来查看相关会话信息。

在信息窗格中，顶部区域包含一些控件的视图，在这里可以搜索显示的文本。如果要使用正则表达式进行搜索，选择 Regex 复选框。状态栏将显示行数和当前行，并找到一个条目的当前列以及相匹配的条目总数。

### 2. 会话快捷菜单

右键单击会话显示一个快捷菜单，允许用户调查选定会话。命令的可用性取决于所选的会话。

用户可以很容易地导出一个文本或 XML 文件中的每个会话的 URL、评论、隐藏字段和各种其他信息的列表。

### 3. 编辑漏洞

WebInspect 评估应用程序的漏洞后，用户可能需要为各种各样的原因编辑并保存结果，包括：

- 安全性(Security)：如果一个 HTTP 请求或响应中包含密码、账号或其他敏感数据，用户可能想删除或修改此信息，使扫描结果在组织中对其他人可用。
- 校正(Correction)：WebInspect 偶尔会报告“误报”，这种情况发生在 WebInspect 检测有可能是漏洞迹象时，但进一步的调查由开发人员确定实际上是否为误报。用户可从会话中删除该漏洞或删除整个会话。或者，可将它指定为误报(右键单击任一网站或序列视图中的会话，然后选择标记为误报)。
- 严重性修改(Severity Modification)：如果不同意漏洞的等级，可指定一个不同等级，使用以下规模：

| | |
|---|---|
| 0 - 9 | Normal |
| 10 | Information |
| 11 – 25 | Low |
| 26 - 50 | Medium |
| 51 - 75 | High |
| 76 - 100 | Critical |

- 记录保存(Record Keeping)：用户可修改任何与个人有关的报告字段的漏洞(摘要、执行、建议、实施、修正和参考资料)。例如，用户可添加一个段落来描述如何真正解决问题的修正部分。
- 增强(Enhancement)：如果发现一个新漏洞，可以定义它，并把它添加到一个会话作为一个自定义的漏洞。

按照下面的步骤编辑会话：

1) 在导航窗格中，右键单击包含一个漏洞的会话或在摘要窗格中右键单击一个 URL。

2) 从快捷菜单中选择 Edit Vulnerability(编辑漏洞)。编辑漏洞窗口如图 7-25 所示。

3) 选择一个漏洞(如果会话包含多个安全漏洞)。

4) 要添加一个现有的漏洞会话(也就是，一个存在于数据库中的漏洞)，单击 Add Existing(添加现有)。

a) 在添加现有的漏洞窗口中，输入一个漏洞名称的一部分，或一个完整的漏洞 ID 号或类型。

b) 单击 Search(搜索)。

c) 选择一个或更多的搜索的漏洞。

d) 单击 OK(确定)。

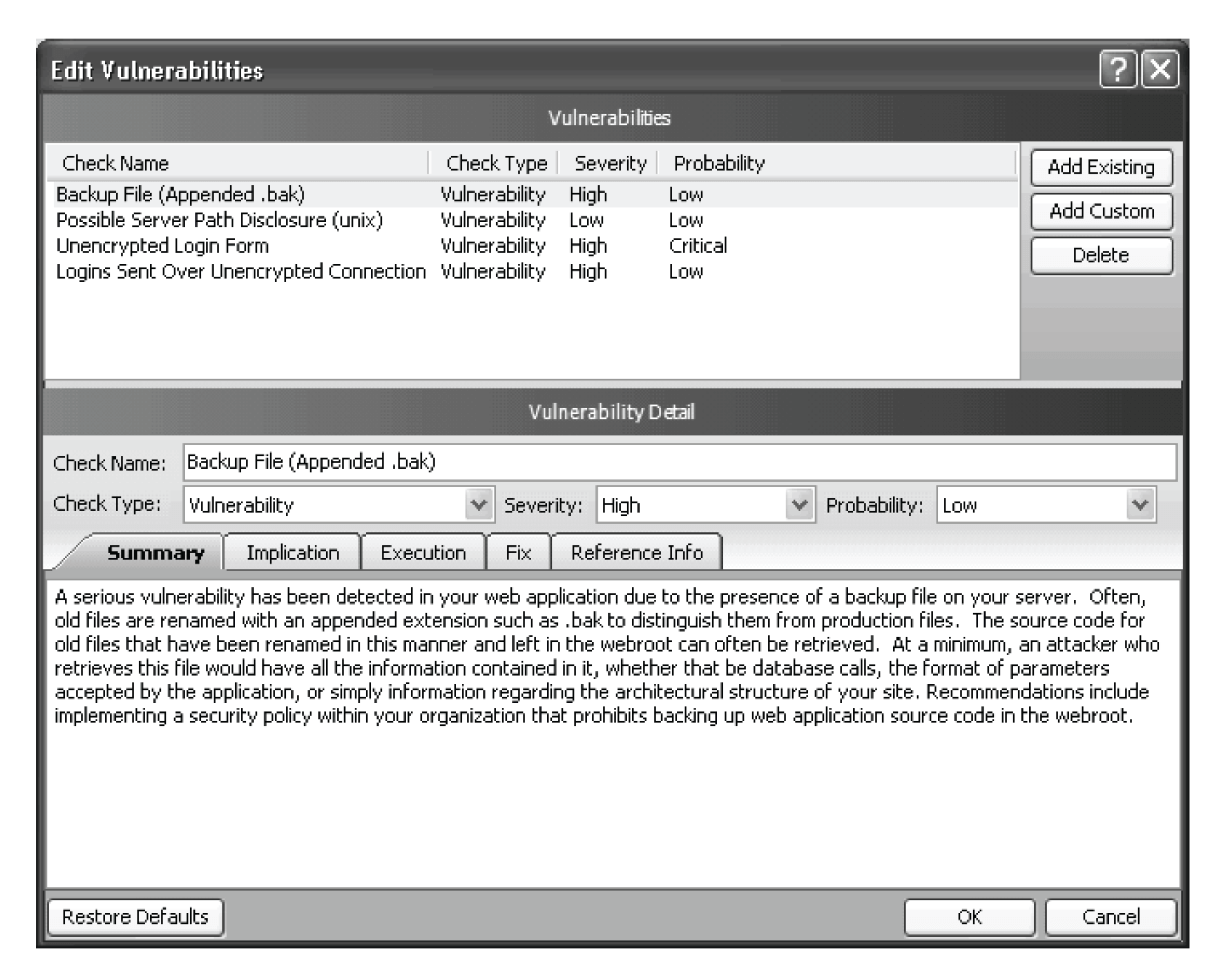



图 7-25　编辑漏洞窗口

5) 要添加一个自定义漏洞，单击 Add Custom(添加自定义)。然后，可以编辑该漏洞，正如在步骤 7 中所描述。

6) 要删除选定会话的漏洞，单击 Delete(删除)。

7) 要修改漏洞，从漏洞详细部分选择不同的选项。用户也可以修改出现在摘要的描述、含义、执行、修复、参考信息选项卡的漏洞。

8) 单击 OK(确定)保存更改。

## 7.2.9　检查结果：Web 服务扫描

大多数 Web 服务使用简单对象访问协议(SOAP)，在 Web 服务器和客户端 Web 应用程序之间发起信息请求来发送 XML 数据。XML 提供了一个框架描述，并包含结构化数据。客户端的 Web 应用程序可以容易地理解所返回的数据，并将该信息提供给最终用户。

访问 Web 服务客户端的 Web 应用程序接收到一个 Web 服务定义语言(WSDL)文件，以便它知道如何与该服务进行通信。WSDL 文档描述的程序包含在 Web 服务器中，如图 7-26 所示。

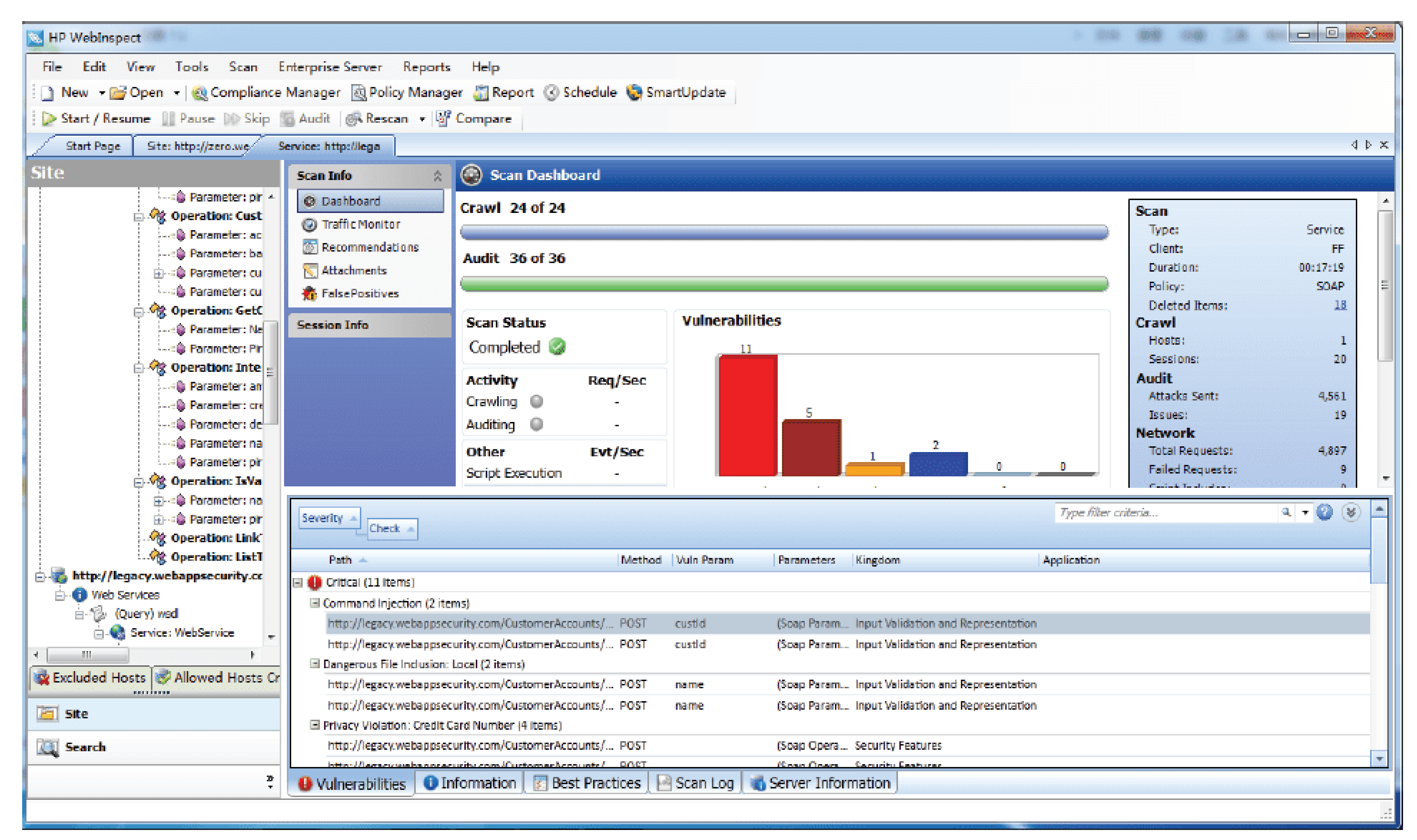



图 7-26　Web 服务扫描检查结果

在导航窗格中或摘要窗格中的漏洞选项卡上选择一个会话对象后，用户可从会话信息面板选择该选项。

## 7.2.10　导出扫描

WebInspect 自动在应用程序设置的指定目录中保存扫描结果。

使用导出功能将扫描信息传送到另一个 WebInspect 应用程序或评估管理平台(AMP)。

按照下面的步骤来导出扫描：

1) 打开要导出的扫描或单击包含一个打开的扫描的选项卡。单击 File→ Export→ Scan(“文件”→“导出”→“扫描”)或 File→ Export→ Scan to Software Security Center(“文件”→“导出”→“扫描到软件安全中心”)，或在开始页面的管理扫描窗格中选择一个扫描，单击 Export(导出)按钮的下拉箭头，然后选择 Export Scan(导出扫描)或 Export Scan to Software Security Center(导出扫描到软件安全中心)。

导出扫描窗口(或导出扫描到软件安全中心窗口)会出现，如图 7-27 所示。

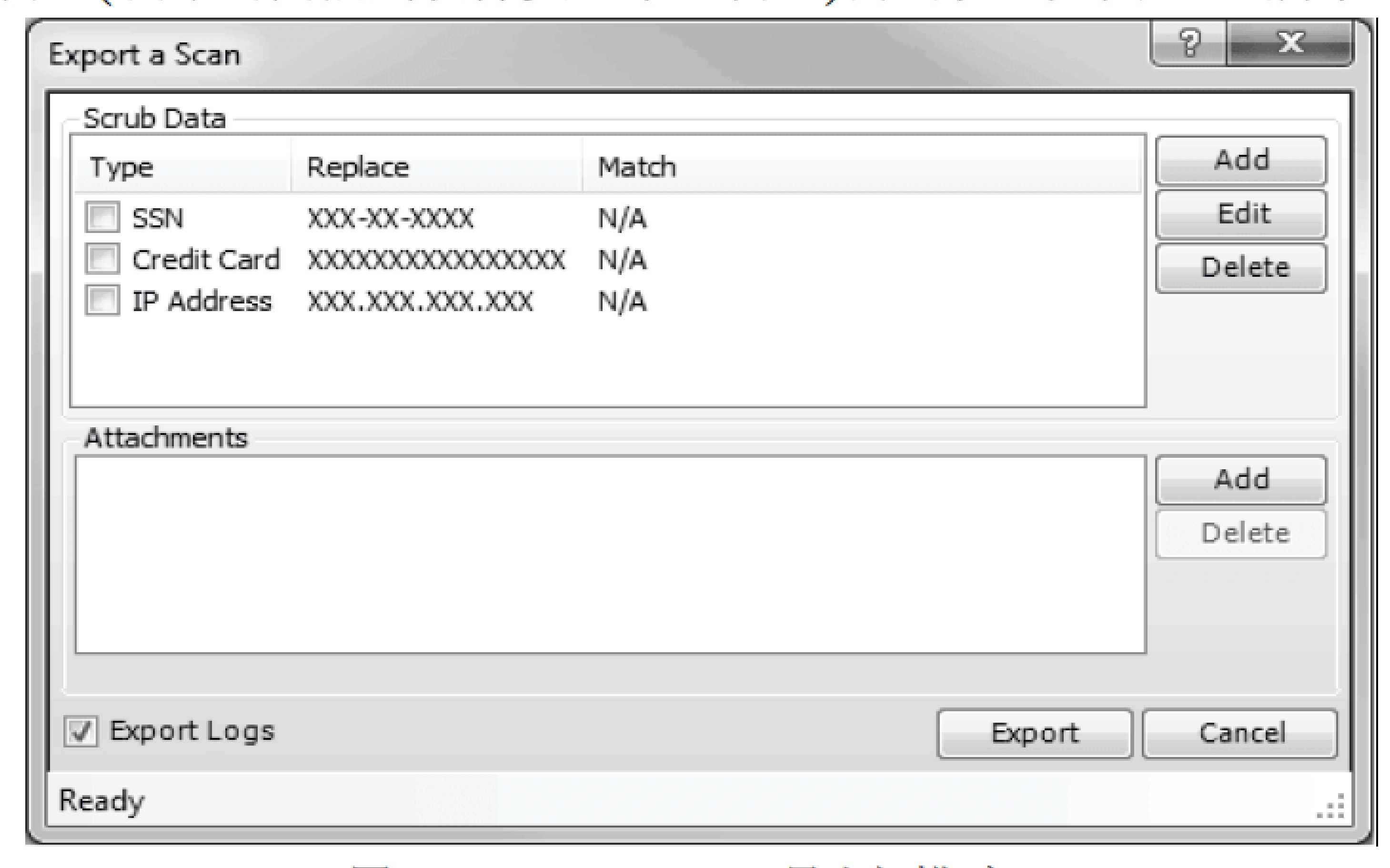



图 7-27　Export a Scan(导出扫描)窗口

2) 擦洗数据(The Scrub Data)组在默认情况下包含三个不可编辑的正则表达式函数替代 X 中每个数字的格式化字符串，如一个社会安全号码、信用卡号码或 IP 地址。如果要包含一个搜索和替换功能，需选择与其相关的复选框。

3) 要创建擦洗功能：

a) 单击 Add(添加)。

b) 在添加擦洗输入窗口中，选择 Regex(正则表达式)或从 Type(类型)列表中选择 Literal(文字)。

c) 在 Match(匹配)框中，输入要查找的字符串(或一个字符串表示的正则表达式)。如果使用正则表达式，用户可单击浏览按钮…，打开正则表达式编辑器，使用它可以创建和测试正则表达式。

d) 在 Replace(替换)框中，输入将由匹配字符串替换的目标指定字符串。

e) 单击 OK(确定)。

| 注意 | Publish Scan to Software Security Center 窗口不包含附件或日志的选项。如果要导出到软件安全中心，请转到步骤 6。 |
|---|---|

4) 如果要导出到软件安全中心，请转到步骤7。

5) 如果想要包含一个附件：

a) 在 Attachments(附件)组中，单击 Add(添加)。

b) 使用标准文件选择窗口，导航到包含要附加文件的目录。

c) 选择一个文件，然后单击 Open(打开)。

6) 如果要包括扫描的日志文件，选择 Export Logs(导出日志)。

7) 单击 Export(导出)。

8) 使用标准的文件选择窗口，选择一个位置，然后单击 Save(保存)。

### 7.2.11　导出扫描详细信息

使用导出功能保存 WebInspect 爬行或审计时收集到的信息。

1) 打开扫描，或单击包含扫描的选项卡。

2) 单击 File→ Export→ Scan Details(“文件”→“导出”→“扫描详细信息”)。出现 Export Scan Details(导出扫描详细信息)窗口，如图 7-28 表示。

3) 从 Details(详细信息)信息列表中，选择要导出信息的类型。可用条目取决于扫描进行的类型。

4) 从 Export Format(输出格式)列表中选择一种格式(文本或 XML)。

### 7.2.12　发布到软件安全中心

#### 1. 通过 WebInspect Enterprise 发布

通过 WebInspect Enterprise，请使用以下过程将扫描数据传输到 HP Fortify Software 安全中心服务器。

1) 配置 WebInspect Enterprise 和软件安全中心。

2) 运行扫描中的 WebInspect(或使用导入或下载扫描)。

3) 单击 Enterprise Server 菜单，并选择 Connect to WebInspect Enterprise(连接 WebInspect Enterprise)。系统将提示提交的凭证。

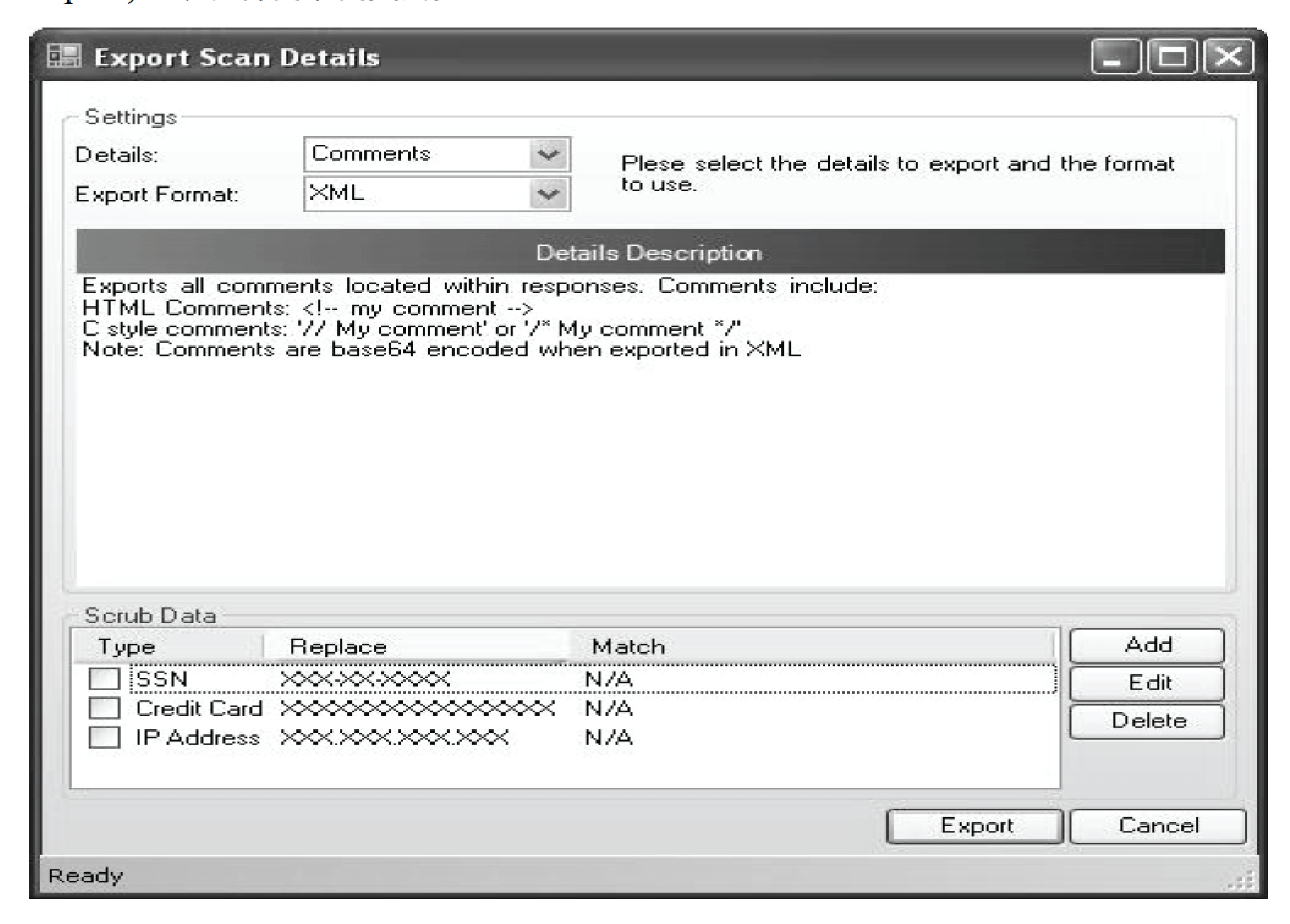



图 7-28　Export Scan Details(导出扫描详细信息)窗口

4) 在具有焦点的选项卡上，可以发布扫描：

a) 单击 Synchronize。

b) 选择一个项目和版本，然后单击 OK(确定)。

c) 检查结果。在摘要窗格中列将出现指定“发布状态”(Published Status)和“待定状态”(Pending Status)。发布状态是这个漏洞扫描发布到 WebInspect Enterprise 上一次的状态。待定状态是扫描公布之后，漏洞将是什么状态。根据不同的待定状态，可以修改它来指定该漏洞是否已经解决或依然存在(见下 7 步)。此外，命名为“未找到”的一个新选项卡出现；此选项卡包含以前扫描中检测到的漏洞，但并不是当前扫描中的漏洞。用户可以添加屏幕截图和评论漏洞或标记漏洞为误报或忽略。还可以查看并重新测试漏洞，修改扫描结果，直到准备好发布。

d) 单击 Publish。转到步骤 7。

5) 要从扫描列表中进行选择，可执行以下步骤：

a) 单击 Enterprise Server 菜单，然后选择 Publish Scan(发布扫描)。

b) 在 Publish Scan to Software Security Center 对话框中，选择一个或多个扫描。

c) 选择一个项目和项目版本。

d) 单击 Next(下一步)。WebInspect 可以使用 SSC 自动同步。

6) WebInspect 列出要公布的漏洞数量，按状态和严重程度进行分类。为确定身份，

WebInspect 可在当前扫描中比较以前提交的漏洞(通过 SSC 同步获得)。如果这是第一次扫描提交的项目版本，所有漏洞将是“新漏洞”。

如果一个漏洞被先前报告过，但不在当前扫描中，它会标记为“未找到”。用户必须确定它是否因为已被固定或扫描配置不同而未被发现(例如，可能使用不同的扫描策略，或者扫描网站的不同部分，或者提前终止扫描)。

7) 如果扫描是为了回应 HP Fortify Software 安全中心，选择 Associate scan with an “In Progress” scan request for the current project version(在当前项目的版本关联“进行中”的扫描请求)。

8) 单击 Publish(完成)。

| 注意 | 单击 Publish 后，扫描排队等待上传到 WebInspect Enterprise。它被上传后，WebIns pect 可将企业排队扫描发布到共享服务中心。出于这个原因，发布在待定状态和发布状态列中的结果不会立即显示出来。如果要监视上传和发布操作，可使用 WebInspect Enterprise WebConsole(WebInspect EnterpriseWeb 控制台)按钮连接到 WebInspect Enterprise。 |
|---|---|

### 2. WebInspect Enterprise 集成化

HP Fortify Software 安全中心(SSC)是一套紧密集成的解决方案，用于识别优先级分配和修复安全漏洞软件。它使用 HP Fortify 静态代码分析器进行静态分析，使用 HP WebInspect 进行动态应用安全测试。WebInspect Enterprise 提供了一个中央位置管理多个 WebInspect 扫描和关联扫描结果，在 SSC 内可以直接发布到个人项目版本。

经过 WebInspect 扫描，与 WebInspect Enterprise 进行同步来获取历史信息，比较那些历史上扫描的漏洞，然后分配一个状态给每个漏洞，如表 7-9 所示。

表 7-9　SSC 说明

| SSC 状态 | 描　述 |
|---|---|
| 新建 | 以前未报告的问题 |
| 现有 | 在扫描历史上已经存在的漏洞 |
| 未发现 | 在扫描历史上未发现的漏洞。这可能是因为(a)该漏洞已经被修复并不再存在，或(b)因为最新的扫描中使用不同的设置，或扫描该网站的不同部分，或者由于其他原因未发现漏洞 |
| 已解决 | 已修复漏洞 |
| 重新引入 | 出现在当前扫描的漏洞，但以前报道 Solved(已解决) |
| 仍然是一个问题 | 这是在当前扫描“未找到”的一个漏洞，事实上存在 |

要更改单个漏洞的 SSC 状态，请右键单击 Vulnerability(漏洞)选项卡上的漏洞，然后选择 Modify Publish Status(修改发布状态)。连接到 WebInspect Enterprise 和已同步 WebInspect 软件安全中心后此选项才会出现。

下面的例子演示了将安全漏洞集成到 HP Fortify Software 安全中心假想的一系列扫描。

- 第 1 次扫描

1) 扫描目标网站的 WebInspect。在这个例子中，假设只有一个漏洞(漏洞号 A)被发现。

2) 检查结果。可添加屏幕截图和评论漏洞或标记漏洞为误报或忽略。还可以检查，重新测试，并删除漏洞。

3) 扫描与项目版本 SSC 同步，然后发布扫描。

- 第 2 次扫描

1) 第 2 次扫描再次显露漏洞号 A，但也发现了 4 个漏洞(漏洞号 B、C、D 和 E)。

2) 扫描项目版本在 SSC 中同步。

3) 检查结果。如果发布第一次扫描时，对漏洞号 A 加入审计数据(如评论和截图)，数据将被导入到新的扫描。

4) 发布扫描到 SSC。漏洞号 A 将被标记为“现有”，漏洞号 B、C、D、E 将被标记为“新建”，5 个项目将存在于 SSC 系统。

- 第 3 次扫描

1) 第 3 扫描发现漏洞号 B、C 和 D，而不是漏洞号 A 或漏洞号 E。

2) 扫描的项目版本在 SSC 中同步。

3) 重新测试漏洞号 A 后，确定它事实上存在。更改其待定状态为 Still a Problem(仍然是一个问题)。

4) 重新测试漏洞号 E 后，确定不存在。更改其待定状态为 Solved (已解决)。

5) 发布扫描到 SSC。漏洞号 B、C 和 D 将被标记为 Existed(现有)。5 个项目将存在于 SSC 系统。

- 第 4 次扫描

1) 第 4 次扫描没有发现漏洞号 A 或 B。扫描发现漏洞号 C、D、E 和 F。

2) 扫描的项目版本在 SSC 中同步。

3) 漏洞号 E 先前宣布要解决，所以它的状态设置为 Peimport(重新引入)。

4) 检查未发现这个漏洞(A 和 B)。如果确定该漏洞依然存在，更新待定状态为 Still a Problem(仍然是一个问题)。如果重新测试验证该漏洞不存在，更新的待定状态为 Solved(已解决)。

5) 发布扫描到 SSC。漏洞号 C 和 D 仍然标有 Existed(现有)。

### 3. 通过 AMP 发布

连接到 AMP，使用下列步骤将扫描数据发布到 HP Fortify Software 安全中心：

1) 单击 WebInspect Enterprise Server 菜单，然后选择 Publish Scan(发布扫描)。

2) 在 Publish Scan to Software Security Center 窗口中，从扫描名称列选择 WebInspect 扫描。

| 注意 | 要在不同的数据库访问扫描，单击 Connections(连接)，然后在数据库应用程序设置 Connection Settings for Scan Viewing(扫描视图的连接设置)下更改选项。 |
|---|---|

3) 从 AMP 组的下拉列表中选择一个网站。可以通过选择 Show Organizations(显示组织)查看组织和项目。

每次扫描都必须分配到一个网站。该计划试图选择匹配扫描文件中的“扫描网址”的网站，但可以选择一个替代。

如果相应网站不存在，可在 AMP 网站使用下列程序创建。必须具有 AMP 权限才能创

建一个网站。

a) 单击 Create Site(创建网站)(在对话框的右上角)。

b) 为对 AMP 服务器窗口建立网站，提供以下信息。

项目(Project)：选择项目名称。

网站名称(Site Name)：输入该网站的名称。

网址(URL)：输入一个完全限定的域名或 IP 地址。

阶段(Phase)：输入阶段的名称，或从阶段列表中选择一个现有的名称(可选)。对于分配了阶段的网站，可只显示这些网站是一个特定阶段的成员。

组(Group)：输入组的名称，或从组列表中选择一个现有的名称(可选)。如果创建网站群，可以只显示这些网站是一个特定组的成员。

身份验证(Authentication)：如果需要身份验证，从列表中选择一个类型。

权重(Weight)：权重用来计算风险评分。该网站的风险评分等于最近完成扫描的该网站乘以用户在此处选择的值。它允许用户表明一些网站更重要或比其他网站具有更高的风险。

c) 单击 OK(确定)。

4) 在 Software Security Center(软件安全中心)组中，单击登录。

5) 输入凭据，然后单击 OK(确定)。

6) 选择一个项目版本。

7) 单击 Publish(发布)。

### 7.2.13　HP TippingPoint 导出保护规则

使用导出功能来创建一个.xml 文件，在 Web 应用程序的扫描过程中，基于 HP WebInspect 检测到的漏洞，该文件可以导入到 TippingPoint 数字疫苗工具包。然后，通过入侵防御系统(IPS)设备的一个网络分布，数字疫苗工具包可以创建过滤器并将其导入到 HP TippingPoint 的安全管理系统。

要创建可以导入到 TippingPoint 的数字疫苗工具包文件，可执行以下操作。

1) 打开感兴趣的扫描(或单击包含一个要打开的扫描的选项卡)，然后单击 File(文件)，再依次单击 Export(导出)和 Protection Rules to HP TippingPoint(HP TippingPoint 保护规则)，如图 7-29 所示。

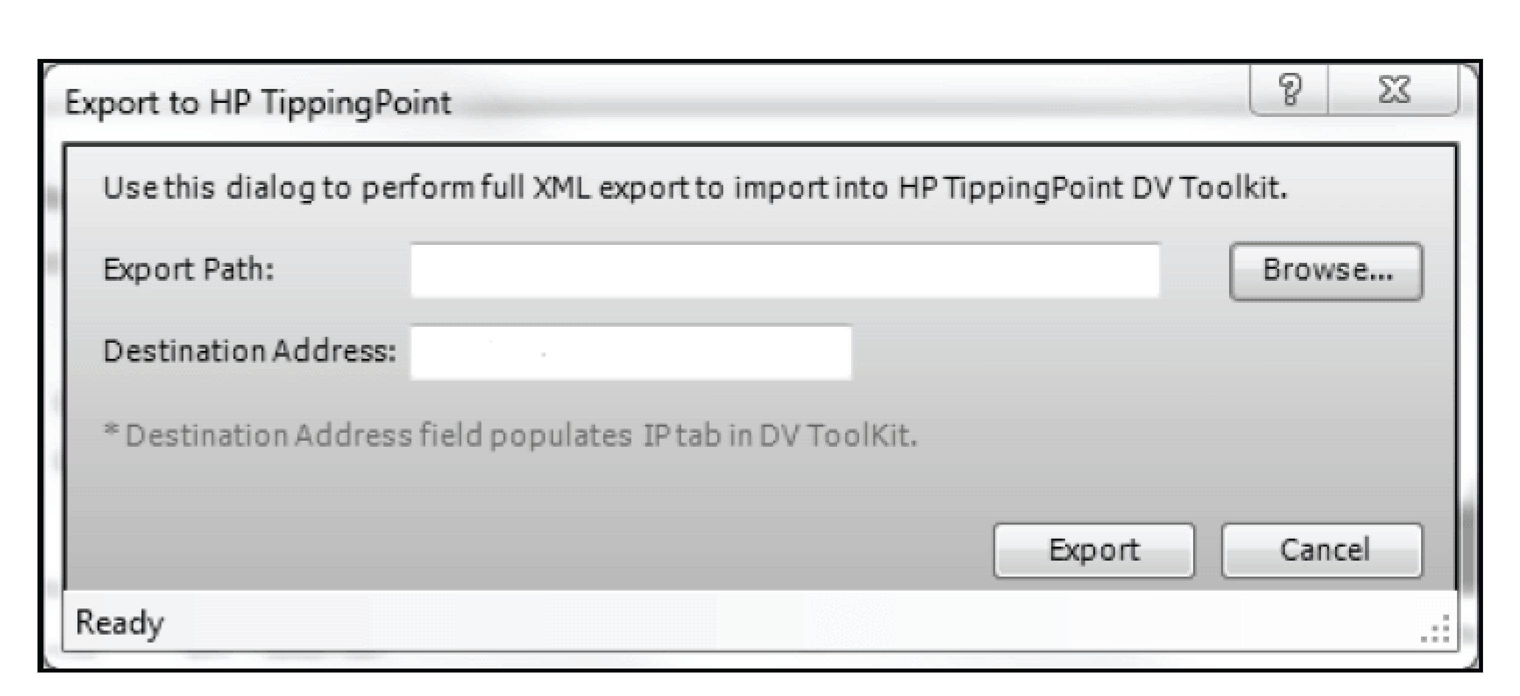



图 7-29　HP TippingPoint 保护规则

2) 指定 Export Path(导出路径)，即该目录中要保存新文件。

3) Destination Address(目的地址)是可选的。

4) 单击 Export(导出)。

映射的漏洞转化并导出到部分或全部指定路径下的文件：

- TpXssRules.xml
- TpSqlinjRules.xml
- TpFileIncludeRules.xml

5) 在 TippingPoint 的数字疫苗工具包，单击 File(文件)，再单击 Import XML(导出 XML)，在同一时间导入每个.xml 文件。忽略所有过滤器验证信息(单击 Yes 继续)。导入文件的内容累积在过滤器之一的.csw 文件。

6) 单击 File(文件)，再单击 Save As(另存为)，累积文件(accumulated file)保存为.csw 文件。

7) 要将.CSW 文件导入到 TippingPoint 的安全管理系统，单击 Profiles(配置)，单击 DV Toolkit Packages(DV 工具包)，然后单击 Import(导入)并选择.csw 文件。导入完成后，单击 Activate(激活)，激活完成后，单击 Import(导入)并选择要分发文件的设备。IPS 配置文件会阻止过滤器。

### 7.2.14　Web 应用防火墙导出保护规则

生成并保存一个完整漏洞的导出文件(.XML)，该漏洞由 HP WebInspect 的 Web 应用程序的扫描过程中检测：

1) 打开感兴趣的扫描(或单击包含一个要打开的扫描的选项卡)，然后单击 File(文件)，再依次单击 Export(导出)和 Protection Rules to Web Application Firewall(Web 应用防火墙保护规则)，如图 7-30 所示。

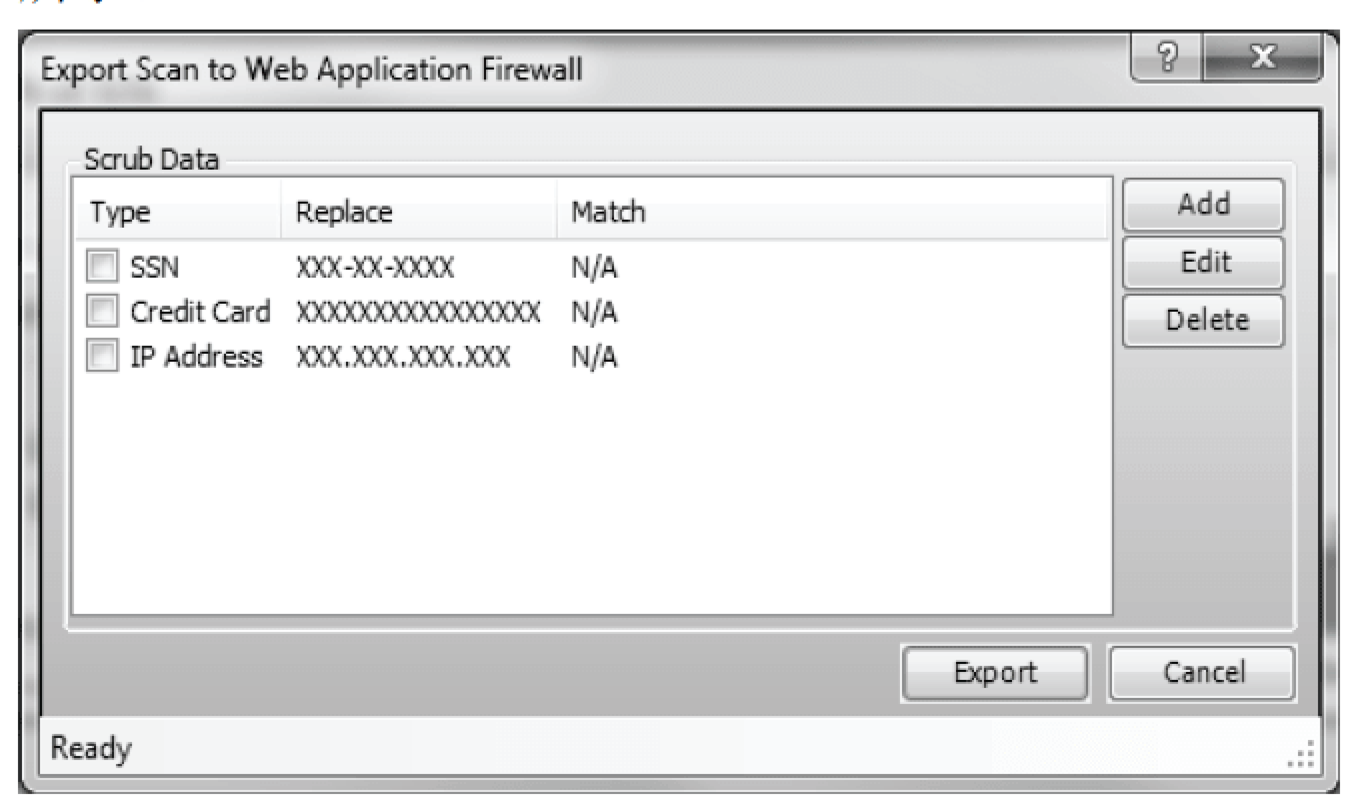



图 7-30　Web 应用防火墙保护规则

2) 指定以同样的方式擦洗数据类型，如 File→Export→Scan(“文件”→“导出”→“扫描”)选项。擦洗数据(Scrub Data)组在默认情况下包含三个不可编辑的正则表达式函数，在字符串格式化中它们将用一个 X 替代每个数字，如一个社会安全号码、信用卡号码或 IP 地址。包括这个搜索和替换函数的数据类型，选择与其相关的复选框。

3) 单击 Export(导出)。

4) 指定要保存导出的数据，然后单击保存路径和文件名。当指定后，一个完整的导出(.XML)文件被保存。

### 7.2.15　导入扫描

遵循以下步骤由另一个实例导入 WebInspect 的扫描。

1) 单击 File(文件)，再单击 Import Scan(导入扫描)。

2) 使用标准文件选择窗口，从 Files of Type(文件类型)列表中选择一个选项：

- Scan files(*.scan)：扫描文件设计创建由 WebInspect 7.0 版本开始。
- SPA files(*.spa)：扫描文件创建在 WebInspect 7.0 版本之前推出。

3) 选择一个文件，然后单击 Open(打开)。

如果附件被扫描导出，这些附件将被导入并保存在导入扫描的子目录。默认位置是 C:\Documents and Settings\<username>\Local Settings\Application Data\HP\HP WebInspect\ScanData\Imports\<DirectoryName>\<filename>，其中 DirectoryName 是导出/导入扫描的 ID 号。

### 7.2.16　运行 AMP 或 WebInspect Enterprise 扫描

此功能专为用户设计，用户更偏爱在 WebInspect 中配置扫描，而非在 AMP 或 WebInspect Enterprise 中。用户可以修改设置，并在 WebInspect 运行扫描，重复这个过程，直到达到用户认定的最佳设置。然后，可将打开的扫描设置为 AMP 或 WebInspect Enterprise，它创建了一个扫描请求；并将它放置在扫描队列中，供下一个可用的检测部件使用。

1) 打开扫描。

2) 如果没有连接到企业服务器，单击 Enterprise Server 菜单，并选择 Connect to AMP or Connect to WebInspect Enterprise(连接到 AMP 或连接到 WebInspect Enterprise)。

3) 单击Scan(扫描)菜单，选择Run in AMP(在AMP中运行)或 Run in WebInspect Enterprise (在 WebInspect Enterprise 中运行)(或直接单击工具栏上相应的按钮)。

4) 如果已连接到 AMP：

a) 在 AMP 窗口运行扫描的 Scan Name(扫描名称)文本框中，键入一个名称，或保留名称的空白，让 AMP 指定名称，如图 7-31 所示。

b) 如果在 AMP 上一个网站存在与扫描对应的 URL，该网站被默认选中。用户可以使用相应的按钮更改网站，也可以创建一个网站。

**注意**　当连接到 8.1 或更高版本时，除了网站名称外，本 Site(网站)列表中还会显示组织和项目名称。对于 AMP 8.0 和更早的版本，只有列出网站名称。

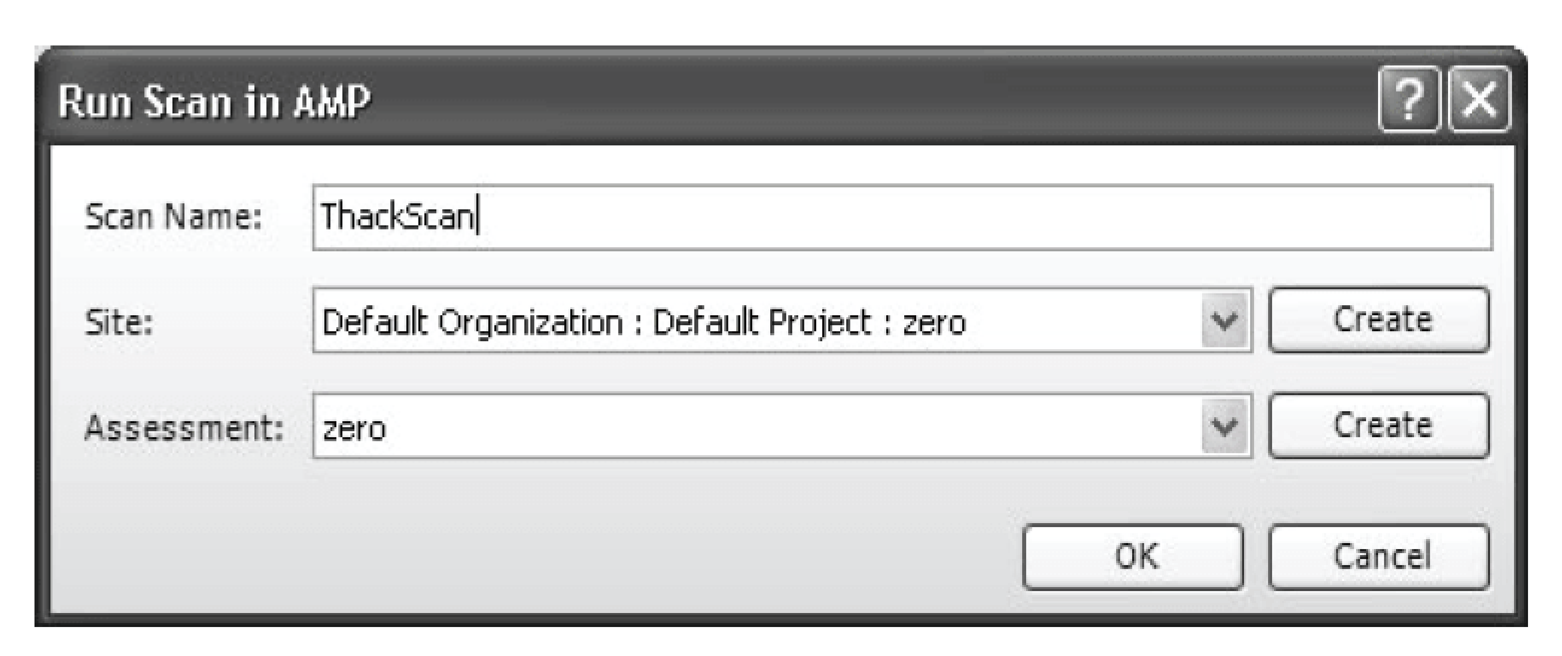



图 7-31　运行扫描的 AMP 窗口

c) 选择将与此扫描相关联的一个评估(可选)，或单击 Create(创建)创建一个评估。

注意　只有 WebInspect 连接到 AMP 9.0 或更高版本，才会启用评估。此外，要建立一个评估，必须拥有 AMP 的管理权限。

d) 单击 OK(确定)。

对于 AMP 8.0 或更早版本，必须有权创建自定义扫描。对于 AMP 8.1 或更高版本，必须有权在项目中创建扫描。

5) 如果已连接到 WebInspect Enterprise：

a) 在 Run Scan in WebInspect Enterprise 对话框中，输入扫描的名称，如图 7-32 所示。

b) 选择一个项目和一个项目版本。

c) 单击 OK(确定)。

如果通过所有的权限检查，扫描被创建，并根据角色(最多 3 个，这是默认的)允许的最高优先级给扫描分配优先级。

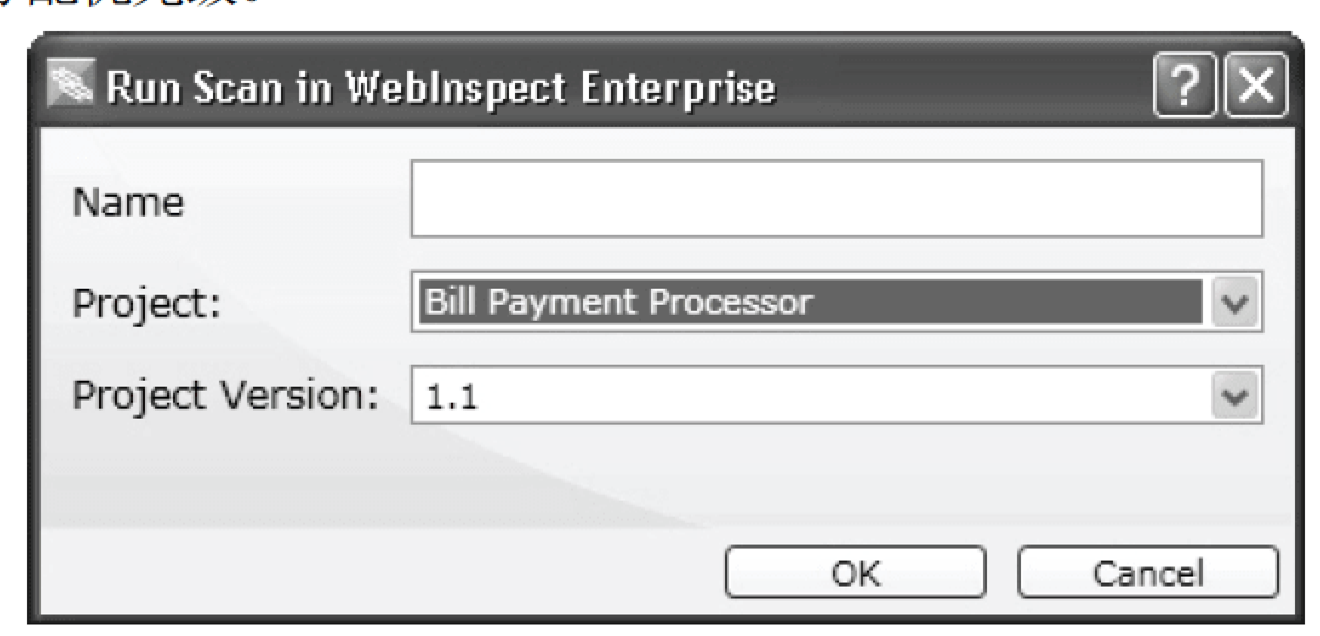



图 7-32　Run Scan in WebInspect Enterprise 对话框

## 7.2.17　上传一个扫描到企业服务器

使用以下步骤从WebInspect上传一个文件扫描到企业服务器(评估管理平台或WebInspect Enterprise)。

1) 单击 WebInspect Enterprise Server(WebInspect Enterprise 服务器)菜单，然后选择 Upload Scan(上传扫描)。

2) 在上传扫描窗口中，选择扫描名称列中的一个或多个 WebInspect 扫描。要访问不同的数据库扫描，单击 Connections(连接)，然后在数据库应用程序设置中，在 Connection Settings for Scan Viewing(扫描视图的连接设置)下更改选项。

3) 如果上传到 AMP，从 AMP 的网站列的下拉列表中为每个扫描选择网站。每次扫描都必须分配到一个网站。该计划试图选择匹配扫描文件中的“扫描网址”的网站，但也可以选择另一个。如果相应的网站不存在，用户可以创建一个(需拥有 AMP 权限)。

4) 从评估栏的下拉列表中选择一个评估(如果上传到 AMP，则是可选的)。

如果一个相应的评估不存在，可以通过以下步骤在 AMP 中建立一个评估。用户必须具有 AMP 权限才能创建评估。请注意，只有当 WebInspect 连接到 AMP 9.0 或更高版本时，此功能才被启用。

a) 单击 Create Assessment(创建评估)。

b) 选择与此评估相关联的网站。

c) 为评估输入一个名称和(可选)描述。

d) 单击 OK(确定)。

5) 如果上传到 WebInspect Enterprise，从相应的下拉列表中为每个扫描选择一个项目和项目版本。

该计划试图选择匹配扫描文件中“扫描网址”的项目和项目版本，但用户也可选择另一个替代。

6) 单击 Upload(上传)。

仅当 WebInspect 连接到 AMP 8.1 及更高版本，才会选中 Show Organizations and Projects (显示组织和项目)复选框。

## 7.2.18　管理设置

此功能允许用户创建、编辑、删除、导入和导出扫描设置文件。

注意：用户也可以载入和保存设置，然后从默认设置窗口恢复出厂默认设置。单击 Edit(编辑)，再单击 Default Scan Settings(预设扫描设置)。

1) 从 WebInspect 编辑菜单选择 Manage Settings(管理设置)。打开管理设置窗口，如图 7-33 所示。

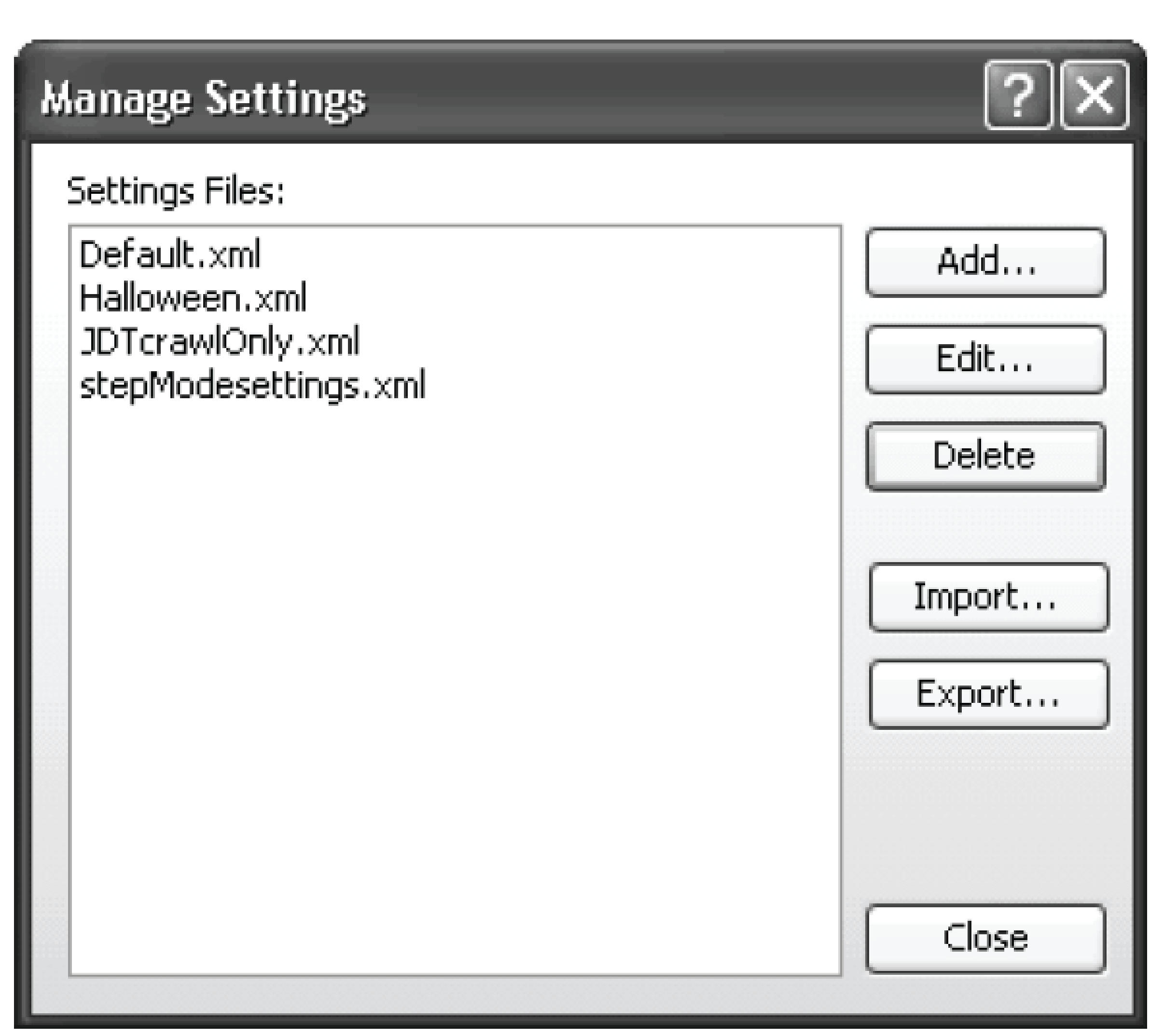



图 7-33　Manage Settings 窗口

2) 要创建一个设置文件：

a) 单击 Add(添加)。

b) 在创建新的设置窗口中，更改设置。

c) 当完成后，单击 OK(确定)。

d) 用一个标准的文件选择对话框，命名并保存文件。

3) 要编辑设置文件：

a) 选择一个文件。

b) 单击 Edit(编辑)。

c) 在创建新的设置窗口中，更改设置。

d) 完成后，单击 OK(确定)。

4) 要删除设置文件：
   a) 选择一个文件。
   b) 单击 Delete(删除)。
5) 要导入一个设置文件：
   a) 单击 Import(导入)。
   b) 使用一个标准文件选择对话中，选择一个设置文件并单击 Open(打开)。
6) 要导出一个设置文件：
   a) 选择一个文件。
   b) 使用一个标准文件选择对话框，选择一个设置文件并单击 Save(保存)。

要使用保存的设置文件扫描：

1) 从 WebInspect 编辑菜单中，选择 Default Settings(默认设置)。
2) 在左侧栏中默认设置窗口的底部，单击 Load settings from file(从文件加载设置)。
3) 使用一个标准文件选择对话框，选择要使用的设置文件，单击 Open(打开)。选择的文件就是默认设置文件。

### 7.2.19　管理扫描

此功能允许打开、重命名或删除包含以前扫描结果的文件。也可以选择一个数据库：本地(扫描在用户机器上配置了 SQL Server 2008 Express 数据库列表)或远程(扫描在服务器上或网络上配置了 SQL Server 2008 Standard 数据库列表)，或选择两者。在开始页面选项卡上，单击 Manage Scans(管理扫描)，如图 7-34 所示。

Home　Manage Scans　Manage Schedule

图 7-34　管理扫描

扫描列表会出现在开始页面右侧窗格中。

默认情况下，WebInspect 在用户机器上列出所有存储在 SQL Server Express 和 SQL Server 标准版(如果配置)的扫描。扫描的当前状态显示在状态栏中。

用户根据喜好可以基于列标题的分类对扫描进行分组(可选)。只需拖动标题，并将其放到分组区域即可。

用户也可以执行大部分功能，要生成一个报告，通过右键单击一个条目，然后从快捷菜单中选择命令。

### 7.2.20　管理计划扫描

用户可以指示 WebInspect 在一个指定的时间或日期进行评估。选择选项和设置保存在一个特殊文件中，通过 Windows 服务器启动 WebInspect(如有必要)，并启动扫描访问。

注意：计划扫描完成后，不会在 WebInspect 开始页面的最近扫描列表中显示。要在完成后访问计划扫描，选择开始页面，然后单击 Manage Scans(管理扫描)。

1) 在开始页面选项卡上，单击 Manage Schedule(管理计划)，如图 7-35 所示。

图 7-35　管理计划

WebInspect 列出所有的计划扫描，包括那些已经运行的计划扫描。

2) (可选)用户可以根据喜好对计划扫描在基于列标题的分类上进行分组。只需拖动标题，并将其放到分组区域即可。

3) 使用工具栏按钮来执行下列功能，如图 7-36 所示。




图 7-36　工具栏按钮

## 1. 计划一个扫描

1) 执行下列操作之一：

- 在 WebInspect 开始页面，单击 Manage Schedule(管理计划)。
- 单击 File(文件)，再单击 Schedule(计划)。

计划扫描列表会显示在开始页面的右窗格中，如图 7-37 所示。

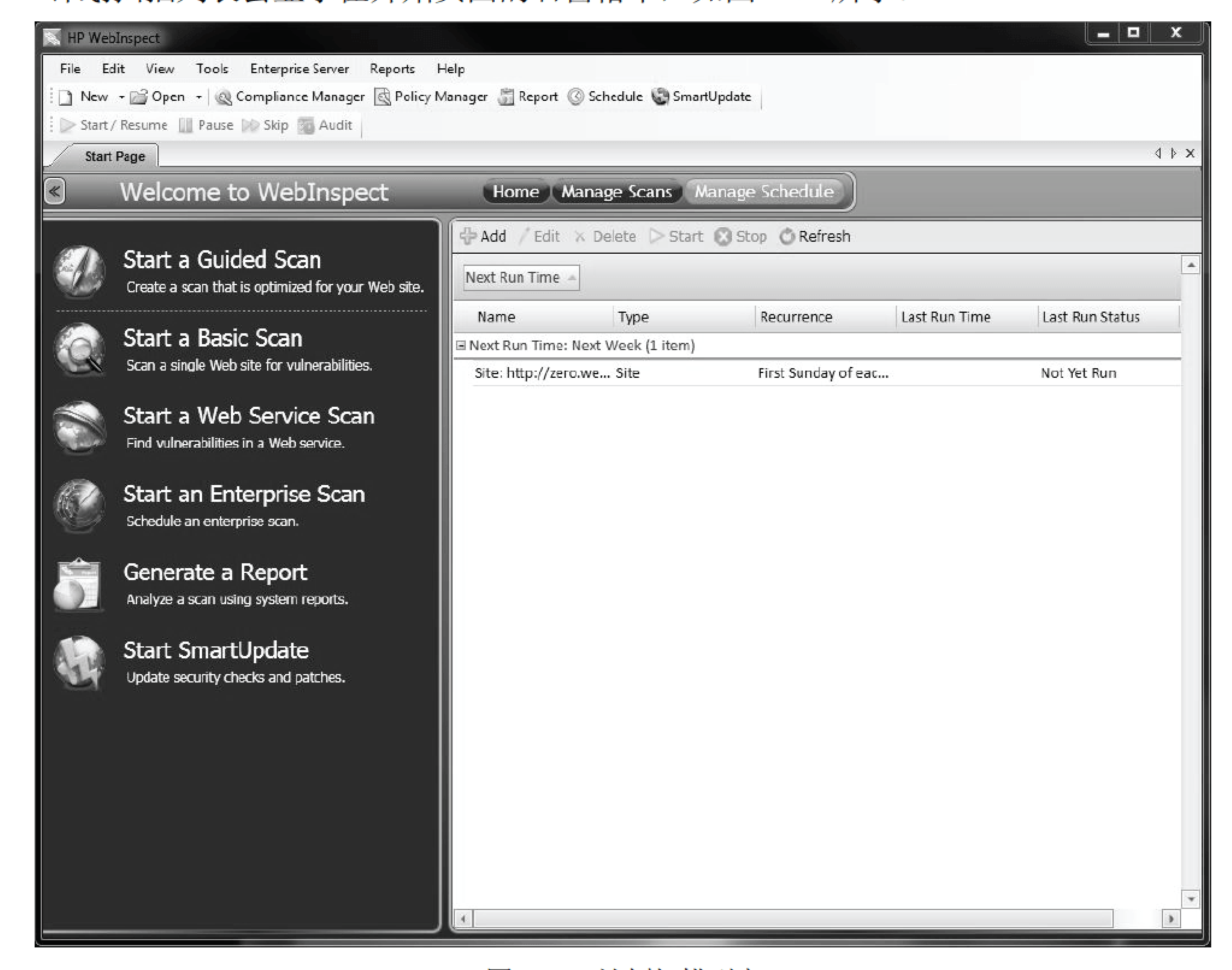



图 7-37　计划扫描列表

2) 单击 Add(添加)。

3) 在扫描组的类型，请选择下列操作之一：

- 网站扫描
- Web 服务扫描
- 企业扫描

4) 指定希望进行的扫描。选项包括：

- 立即(Immediately)。
- 运行一次(Run Once)：当扫描开始时修改日期和时间。用户可以单击下拉箭头以便在日历中选择日期。
- 重复计划(Recurrence Schedule)：使用滑块来选择一个频率(每日、每周或每月)。然后指定扫描开始的时间(对于每周或每月)以及提供其他计划信息。

5) 单击 Next(下一步)。

6) 选择扫描类型的输入设置。

7) 仅对网站和 Web 服务扫描，用户可选择在扫描结束后生成报告：

- 选择 Generate Reports(生成报告)，然后单击 Select a Report 超链接。
- 继续选择报告(下一步中)。

8) 计划扫描不生成报告，单击 Schedule(计划)。

## 2. 选择一个报告

如果选择计划扫描中包括一个报告，将显示 Select a Report 对话框，如图 7-38 所示。

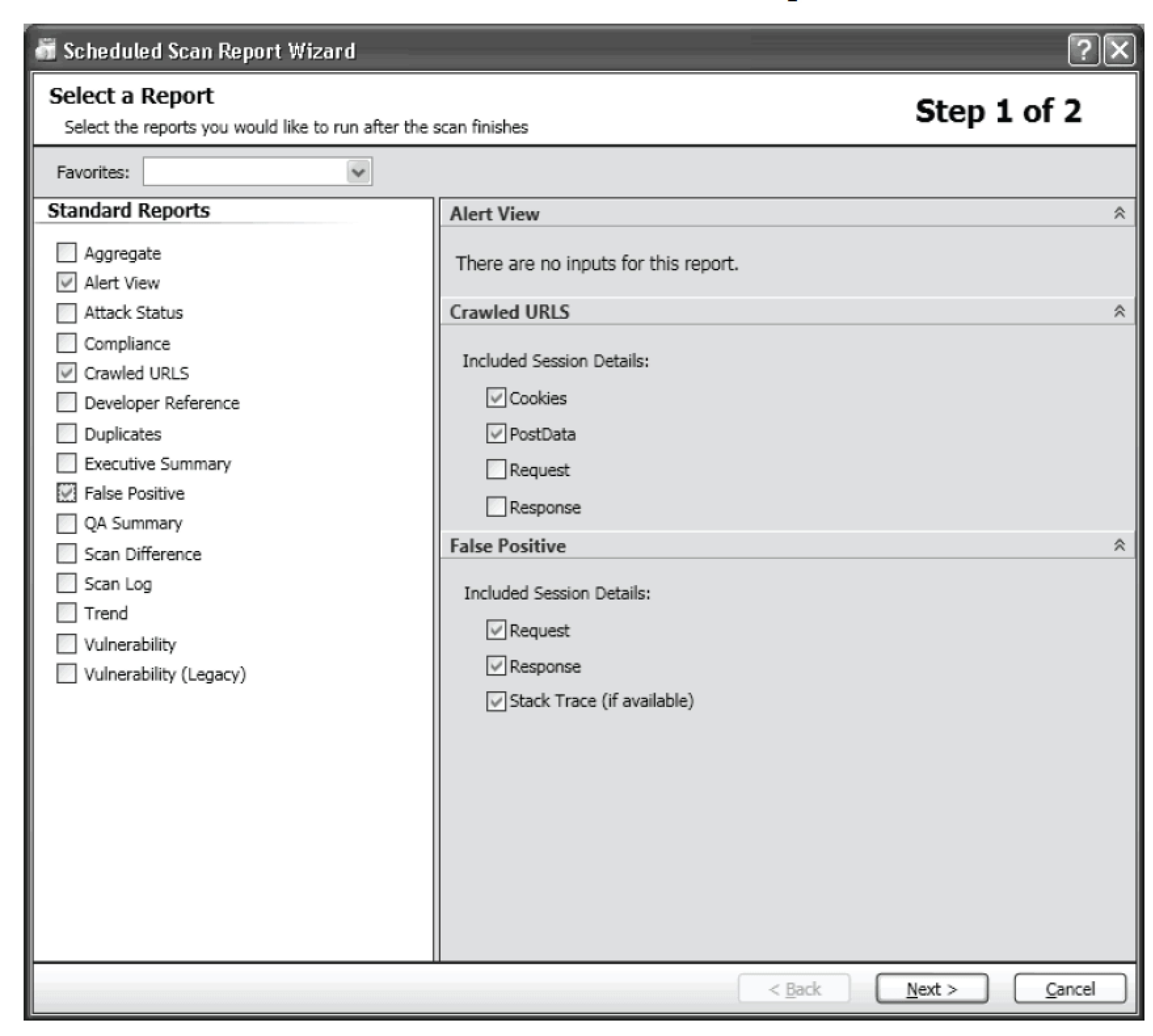



图 7-38　Select a Report 对话框

1) 从收藏夹列表选择报告(可选)。

一个“收藏夹”仅是一个或多个报告及其相关参数的命名集合。要创建一个最喜欢的选定报告和参数，单击 Favorites(收藏夹)列表，然后选择 Add to favorites(添加到收藏夹)。

2) 选择一个或多个报告。

3) 对任何参数提供的信息都是必需的。必需的参数用红色标出。

4) 单击Next(下一步)。

出现 Configure Report Settings 对话框，如图 7-39 所示。

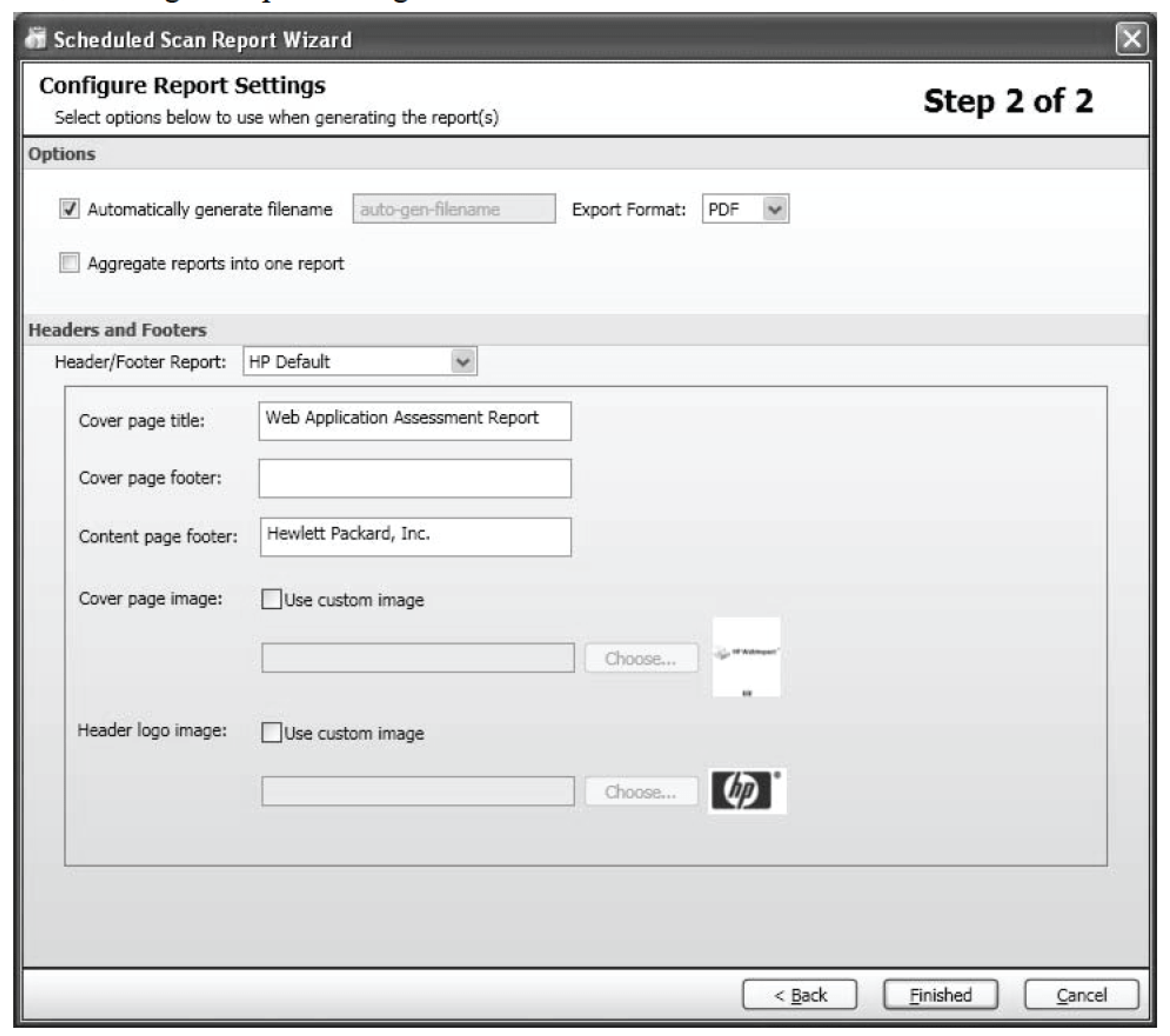



图 7-39　Configure Report Settings 对话框

1) 如果选择 Automatically Generate Filename(自动生成文件名)，该报告文件的名称将被格式化为<reportname><date/time>.<extension>。例如，如果创建 PDF 格式的合规报告，并在 4 月 5 日 6:30 生成报告，文件名是“合规报告 04_05_200906_30.pdf”。这对经常性的扫描，非常有用。

报告被写于应用程序设置生成报告的指定目录中。

2) 如果没有选择 Automatically Generate Filename(自动生成文件名)，在文件名框中输入文件名。

3) 从 Export Format(输出格式)列表中选择报告格式。

4) 如果选择了多个报告，则可以通过选择 Aggregate reports into one report(汇总报告合并为一个)来汇总报告。

5) 选择用于报告定义的页眉和页脚的模板，如有必要，提供所需的参数。

6) 单击 Finished(完成)。

7) 单击 Schedule(计划)。

要在运行时停止计划扫描，请从管理计划列表中选择扫描，并单击 Stop (或右键单击扫描，然后从快捷菜单中选择 Stop Scan(停止扫描))。

要重新启动一个停止扫描，请从管理计划列表中选择扫描，并单击 Start (或右键单击扫描，然后从快捷菜单中选择 Start Scan(开始扫描))。

### 7.2.21 生成报告

按照下面的步骤对已运行的扫描生成报告。这个步骤并不适用于计划扫描。

1) 单击 WebInspect 工具栏上的 Reports(报告)，然后单击 Generate Report(生成报告)，或在 WebInspect 开始页面上单击 Generate a Report(生成报告)，此时将显示一个报告窗口。

2) 选择一个或更多扫描(通过名称、URL 或 IP 地址指定)。

3) 单击 Advanced(高级)(可选，位于窗口底部)，选择要保存的报告以及页眉和页脚模板选项。

4) 单击 Next(下一步)。

5) 从收藏夹列表选择报告(可选)。

6) 选择一个或多个报告：

- 合计(Aggregate)：此报告可让用户组合多台服务器的扫描结果。它列出漏洞总数，并按严重程度显示相关的条形图，还列出每个服务器的漏洞。
- 警报视图(Alert View)：每次成功的攻击代理和相关漏洞的列表(作为信息显示在摘要窗格中的警报选项卡上)。
- 攻击状态(Attack Status)：列出漏洞 ID 号、检查名称以及所有攻击代理的严重等级。
- 合规性(Compliance)：本报告提供等级来使用户的申请符合某些政府法规或公司指导方针。用户必须首先创建一个合规的模板。
- 爬行网址(Crawled URLs)：对于爬行过程中遇到的每一个 URL，该报告列出了发送的 Cookie 和原始的 HTTP 请求和响应。

注意：如果 WebInspect 无法完成这项报告或者报告生成极为缓慢，修改报告如下：

1) 打开报告设计器。

2) 打开爬行的网址：非唯一子报表。

3) 对于 GroupHeaderRequest 和 GroupHeaderResponse 下方的 RichTextBoxes，MaxLength 属性设置为 4096。

4) 保存报告。

- 开发人员参考(Developer Reference)：每个表单的总计和详细说明，包括 JavaScript、电子邮件、评论、隐藏控制和网站上发现的 Cookie，用户可以选择其中一个或多个引用类型。
- 重复(Duplicates)：该报告包含关于漏洞的信息，由 SecurityScope 进行检测，可以追溯到同一来源；其中首先列出不相关漏洞总数和独特漏洞数量的柱状图对比。
- 内容提要(Executive Summary)：基本统计资料，加上能反映用户漏洞应用程序水平的图表和图形。

- 误报(False Positives)：显示原本归类为漏洞的 URL 信息，但后来用户确定是误报。
- QA 摘要(QA Summary)：含断开的链接、服务器错误、外部链接和超时的所有页面 URL 列表。可选择其中一个或多个类别。
- 扫描差异(Scan Difference)：该报告比较了两种扫描和报告差异，如漏洞、页面和发生在一个网站上文件未找到的回应。
- 扫描日志(Scan Log)：在扫描期间(作为信息显示在摘要窗格的扫描日志选项卡上)由 WebInspect 执行活动的顺序列表。
- 趋势(Trend)：此报告可让用户监控开发团队解决漏洞的进展。例如，用户可以检查初始扫描结果和团队何时开始修复这些问题。然后每周一次，重新扫描该网站和归档结果。量化进度，生成一个趋势报告，分析迄今为止进行的所有扫描结果。该报告包括显示漏洞数量的图表，按严重程度，将每次扫描进行的日期定义在一个时间轴上。重要提示：为确保获得可靠结果，每次扫描时使用相同的策略。
- 漏洞(Vulnerability)：每个漏洞(按严重程度选择)的详细报告和解决方案的建议。
- 漏洞(遗留)(Vulnerability(Legacy))：每个漏洞的详细报告，以及有关的补救建议。

5) 任何参数的提供信息可能会被请求，一个感叹号表示一个必需的参数。

6) 如果想制作单独选项卡(而不是合并在一个选项卡的所有报告)的个别报告，选择 Open Reports in Separate Tabs(在单独选项卡上打开报告)。

7) 单击 Finish(完成)，如图 7-40 所示。

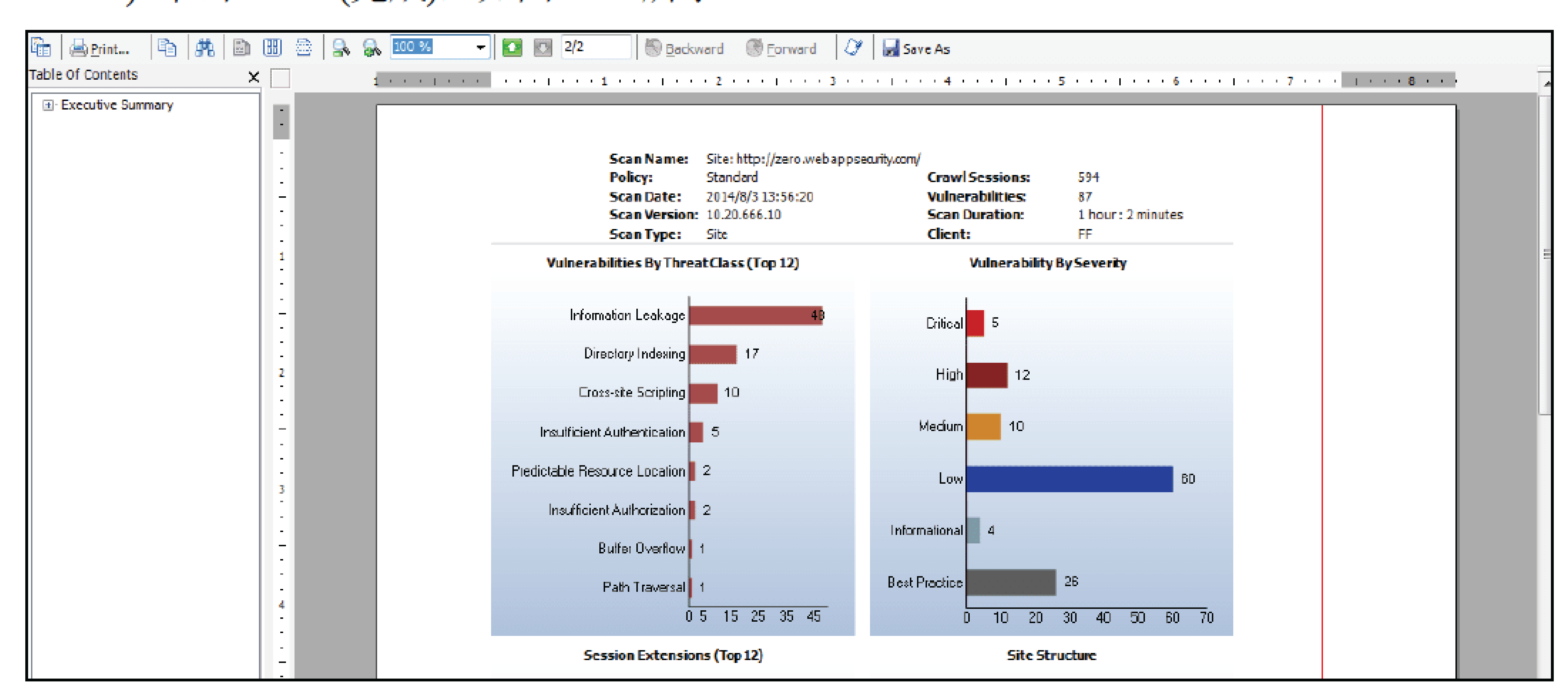



图 7-40　生成的报告

WebInspect 生成报告和显示选项卡后，可以通过单击按钮来保存报告。

报告可保存为下列格式：

- Adobe 便携式数据格式(PDF)
- 超文本标记语言(HTML)
- 本机 WebInspect 的内部格式
- RTF 格式(RTF)
- 文本(TXT)
- 微软的 Excel(XLS)

单击 Advanced(高级)(位于生成报告窗口的底部)，选择保存报告，并设置页眉和页脚模板选项，如图 7-41 所示。

图 7-41　高级报告选项

Save repores to disk (报告保存到磁盘)：选择此选项可将报告保存到磁盘中。

Automatically generate filename (自动生成文件名)：将报告保存到磁盘时选择此选项，该报告文件的名称将被格式化为<reportname><date/time>.<extension>。例如，如果创建 PDF 格式的合规报告，并在 4 月 5 日 6:30 生成报告，文件名是“合规报告 04_05_200906_30.pdf。”这对重复性的扫描非常有用。

如果选择一个以上的报告类型，然后<reportname>将是“联合报告”。

报告存放于应用程序设置生成报告的指定目录中。

如果不选择 Automatically generate filename(自动生成文件名)，用一个文件名替换默认的“自动根文件名”。

导出格式：选择一个报告格式。

页眉和页脚：选择报告的页眉和页脚格式，然后输入或选择组件。

## 7.2.22　许可证管理

首次安装成功并启动 WebInspect 后，系统会提示输入惠普发送给用户的许可证令牌。在输入令牌和客户信息后，应用程序连接 HP 服务器并下载用户特定的安装许可证信息。

有两种方法可以控制 WebInspect 的许可证：

- 直接连接到 HP 公司的许可证服务器。
- 连接到本地许可证和基础设施管理。

### 1. 修改许可证

1) 单击 Edit(编辑)，再单击 Application Settings(应用程序设置)。

2) 选择 License(许可证)类别。

连接到 HP，如图 7-42 所示。

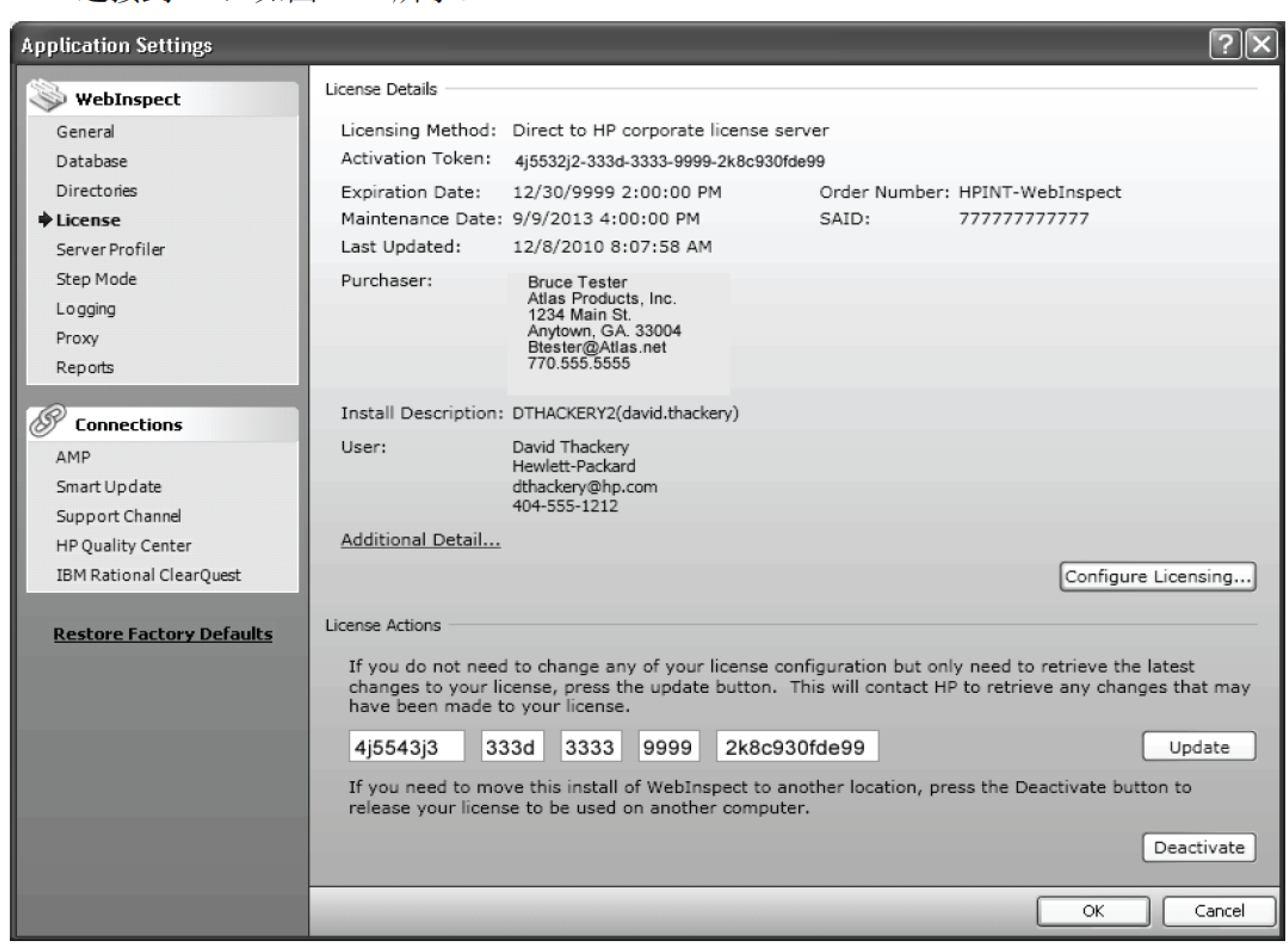



图 7-42　许可证管理界面

### 2. 更新许可证

如果是试用版升级或以其他方式修改许可证，则单击 Update(靠近右下角的超链接)来更新许可证。该应用程序将在计算机上再次连接 HP 服务器和更新本地存储的信息。

### 3. 停用许可证

WebInspect 可以授权被分配到特定的计算机。如果想将许可证转移到其他电脑：

1) 复制激活令牌。

注意：为避免丢失或忘记激活令牌，建议书写或打印保存。

2) 单击 Deactivate(停用)。

3) 在另一台已成功安装 WebInspect 的电脑上，进入 WebInspect 的许可证设置并输入激活令牌。

### 7.2.23　命令行执行

用户可通过使用程序 WI.exe 一个命令行界面启动多个 WebInspect 功能。键入命令时，请使用以下语法，如图 7-43 所示：

```
wi.exe -u url [-s file] [-ws file] [-Framework name] [-CrawlCoverage name][-ps policyID | -pc path]
[-ab|an|am|ad|aa|ak {creds}] [-o|c][-n name] [-e[abcdefghijklmno] file] [-x|xd|xa|xn] [-b filepath] [-v] [-?] [-r
report_name -y report_type -w report_favorite -f report_export_file -g[phacxe][-t
compliance_template_file]] [-d filepath -m filename][-i scanid] [-ir scanid] [-db]
```

图 7-43　使用 WI.exe 命令行界面启动多个 WebInspect 功能

要在命令行中运行多个扫描，创建并执行一个批处理文件，使用类似于下面的代码，如图 7-44 所示：

```
c:
cd \program files\HP\HP Webinspect
wi.exe -u http://172.16.60.19 -ps 4
wi.exe -u http://www.mywebsite.com
wi.exe -u http://172.16.60.17
wi.exe -u http://172.16.60.16
```

图 7-44　运行多个扫描，创建并执行一个批处理文件

内容如表 7-10 所定义，斜体项是要求的值。

表 7-10　命令行语法

| 类　别 | 参　数 | 定　义 |
|---|---|---|
| 通常 | ? | 显示用法 |
| | u {*url*} | URL (include host/port/scheme)<br>注意 1：当使用-s(设置文件)的-u 参数时，一定要指定-x、-xa、-xd、或-xn 参数来限制扫描文件夹。如果不这样做，可能导致在一定条件下无限制的审计。<br>注意 2：如果 URL 中包含一个符号(&)，则必须用引号把 URL 括起来 |
| 通常 | s {*filename*} | 设置文件<br>注意：设置文件中命令行参数优先于值 |
| | o | 仅审计 |
| | c | 仅爬行 |
| 通常 | X | 限制到根文件夹 |
| | xa | 限制到目录和上一级目录 |
| | xd | 限制到目录和子目录 |
| | xn | 忽略“限制到文件夹”规则中引用的设置文件 |
| | n {*name*} | 扫描名称 |
| | b {*filepath*} | 使用给定的 SecureBase 文件 |
| | d {*filepath*} | 将数据库移动到文件路径 |

(续表)

| 类　别 | 参　数 | 定　义 |
| --- | --- | --- |
| 通常 | M {*filename*} | 将数据库移动到文件名 |
| | i (*scanid*) | 扫描识别码(GUID) |
| 审核策略 | ps {*id*} | 使用非自定义策略，相应值的 ID 是：<br>1=标准<br>2=突击<br>3=SOAP<br>4=快速<br>5=安全<br>6=发展<br>7=空白<br>16=QA<br>17=应用<br>18=平台<br>1001=SQL 注入<br>1002=跨站脚本<br>1003=OWASP Top 10 项目<br>1004=所有检查<br>1005=被动 |
| | pc {*path*} | 使用自定义策略，指定完整的路径和文件名，如：<br>C：\MyPolicies\MyCustomPolicy.policy |
| 认证 | ab {*id:pwd*} | 基本模式(用户名和密码) |
| | an {*id:pwd*} | NTLM 模式(用户名和密码) |
| | ad {*id:pwd*} | 摘要模式(用户名和密码) |
| | aa {*id:pwd*} | 自动模式(用户名和密码) |
| | d {*filepath*} | Kerberos 的模式(用户名和密码) |
| | m {*filename*} | 网络宏认证(路径宏) |
| 输出 | Ea {filepath} | 在完整 XML 格式的导出扫描 |
| | eb {filepath} | 在 XML 中导出扫描的详细信息(全) |
| | ec {filepath} | 在 XML 中导出扫描的详细信息(评论) |
| | ed {filepath} | 在 XML 中导出扫描的详细信息(隐藏域) |
| | ee {filepath} | 在 XML 中导出扫描的详细信息(脚本) |
| | ef {filepath} | 在 XML 中导出扫描的详细信息(设置 Cookie) |
| | eg {filepath} | 在 XML 中导出扫描的详细信息(Web 窗体) |
| | eh {filepath} | 在 XML 中导出扫描的详细信息(URL) |
| | ei {filepath} | 在 XML 中导出扫描的详细信息(请求) |
| | ej {filepath} | 在 XML 中导出扫描的详细信息(会话) |
| | ek {filepath} | 在 XML 中导出扫描的详细信息(电子邮件) |

(续表)

| 类　别 | 参　数 | 定　义 |
|---|---|---|
| 输出 | el {filepath} | 在 XML 中导出扫描的详细信息(参数) |
| | em {folderpath} | 在 XML 中导出扫描的详细信息(网络转储) |
| | en {folderpath} | 在 XML 中导出扫描的详细信息(异地链接) |
| | v | 详细 |
| 报告 | r {name}<br>对于多个报告，用分号分隔报告名称。所有报告都将被包含在一个文件中 | 值名称：<br>合计<br>警报视图<br>攻击状态<br>合规性<br>爬行的网址<br>执行摘要<br>误报<br>QA 摘要<br>趋势<br>漏洞概要<br>漏洞(遗留) |
| | y {report_type} | 报告的类型(“标准”或“自定义”) |
| | f {export_path} | 保存报告文件的路径(完整路径和文件名) |
| | gp | 导出为 PDF |
| | gh | 导出为 HTML 文件(ZIP 文件) |
| | ga | 导出为原始报告文件 |
| | gi | 导出为 TIFF 图像 |
| | gc | 导出为富文本文件 |
| | gx | 导出为文本 |
| 报告 | ge | 导出为 Excel 文件 |
| | t{filepath} | 使用指定的合规性模板文件 |

如果不指定策略，WebInspect 将爬行(但不是审计)网站。

如果指定了一个不存在的策略名称，WebInspect 将不进行扫描。

启动命令行的命令：

1) 从 Windows 开始菜单中选择 Run(运行)。

2) 在打开框中键入 cmd。

3) 单击 OK(确定)。

4) 导航到安装 WebInspect 的目录。默认目录为 C：\ Program Files\ HP\ HP WebInspect。

在对长文件名进行妥善处理过程中，因为它出现在任务管理器，是 WI.exe。扫描数据将暂时缓存在工作目录中，然后移动到扫描目录。

仅当参数括在双引号中，可使用连字符的命令行参数(输出文件等)，如下面所示的“导

出路径”命令：

wi.exe -u http://zero.webappsecurity.com -ea "c:\temp\command-line-test-export.xml"

### 7.2.24　卸载 WebInspect

卸载时，用户可选择修复 WebInspect 或从计算机中删除它。

如果选择 Remove(删除)，用户可选择以下一个或两个选项：

- 彻底删除产品(Remove product completely)：删除 WebInspect 应用程序和所有相关文件，包括扫描存储在本地(非共享)SQL 服务器上的数据、设置文件和日志。
- 停用许可证(Deactivate license)：发布 WebInspect 许可证，它允许将 WebInspect 安装在不同的电脑，而且应用数据和文件不会被删除。

## 7.3　扫描一个网站

更新数据库并准备扫描后，用户就可以确定 Web 应用程序的安全漏洞。

按照下面的步骤来开始扫描：

在 WebInspect 菜单栏上，单击 File(文件)，再单击 New(新建)。或在 WebInspect 工具栏上，单击工具栏上的新建下拉箭头。然后选择要进行扫描的类型。选项包括：

- 向导扫描(Guided Scan)
- 基础扫描(Basic Scan)
- Web 服务扫描(Web Service Scan)
- 企业扫描(Enterprise Scan)

### 7.3.1　向导扫描

当启动向导扫描时，计划显示在左窗格中，当用户指定设置扫描后，用户可以很容易地看到计划。

在右窗格中显示每个向导页上的扫描选项，如下所示：

- 步骤 1：指定连接和扫描类型：指定网站进行扫描和验证连接

1) 验证网站(Verify Web Site)

2) 选择扫描类型(Choose Scan Type)

   a) 标准扫描(Standard Scan)。使用 StartURL 开始爬行和/或审计网站。

   b) 工作流扫描(Workflows Scan)。使用预先录制的宏，开始爬行和/或审计网站。

- 步骤 2：登录 - 指定身份验证设置

1) 验证网站(Verify Web Site)

   a) 网络身份验证(Network Authentication)

   b) 客户端证书(Client Certificates)

2) 应用程序身份验证

   a) 选择登录宏(Select Login Macro)

   b) 记录/编辑登录宏(Record/Edit Login Macro)

- 步骤 3：配置主动学习设置：指定优化任务

1) 个人网站的最佳设置(Profile your site for optimal settings)

2) 增强覆盖网站(Enhance coverage of your site)

- 步骤 4：查看设置

1) 最终检阅(Final Review)

  a) 配置详细选项(Configure Detailed Options)

  b) 验证设置和开始扫描(Validate Settings and Start Scan)

### 1. 步骤 1：指定链接和扫描类型

要启动向导扫描，验证连接，并完成网站设置：

1) 在主页上，单击左侧导航窗格中的 Start a Guided Scan(开始向导扫描)。向导扫描指导会在左窗格中出现选择扫描类型页与突出显示的参数。

2) 在 Start URL 框中，键入或选择网站完整 URL 或 IP 地址进行扫描。

3) 要限制一个区域的扫描范围(可选)，选择限制到文件夹复选框，然后从列表中选择以下选项之一：

- 仅目录(Directory only (self))：WebInspect 将仅爬行和/或审计指定的 URL。例如，如果选择了此选项，并指定 www.mycompany/one/two/的 URL，WebInspect 将只评估“两个”目录。
- 目录和子目录(Directory and subdirectories)：WebInspect 开始爬行和/或审计指定的 URL，但不会访问较高目录树中的任何目录。
- 目录和父目录(Directory and parent directories)：WebInspect 开始爬行和/或审计指定的 URL，但不会访问较低目录树中的任何目录。

4) 单击 Verify(验证)。

5) 如果必须通过代理服务器访问目标网站，在主屏幕左下角显示代理设置区域，单击 Proxy(代理)服务器。

当网站或目录结构出现时，已成功连接到 Start URL。

6) 单击 Next(下一步)。在选择扫描类型窗口中选择扫描类型。

扫描类型

1) 选择以下扫描类型之一：

- 标准扫描(Standard Scan)：WebInspect 从目标 URL 开始进行自动分析，这是正常方式开始新的扫描。
- 工作流扫描(Workflows Scan)：如果选择此选项，会出现额外的工作流区域，显示已创建的工作流表。

2) 如果选择 Standard Scan(标准扫描)，然后键入或在开始的 URL 框中选择完整的 URL 或 IP 地址。

3) 如果选择 Workflows Scan(工作流扫描)，工作流部分会出现在左窗格中。

4) 要完成工作流设置，请单击在工作流表如下的任意一项：

- 记录(Record)：打开统一的 Web 宏录制器，允许创建一个宏。
- 编辑(Edit)：打开统一的 Web 宏录制器，加载选定的宏。

- 删除(Delete)：删除选定的宏(但不会从磁盘中删除)。
- 导入(Import)：打开一个标准的文件选择窗口，选择一个以前录制的宏。
- 导出(Export)：打开一个标准的文件选择窗口，允许保存录制宏。在一个宏被选中或记录后，用户可选择性地指定允许主机。

5) 在扫描方法区域，选择扫描方式(包括仅爬行、爬行和审计、仅审计)。

6) 在策略区域，从策略列表中选择策略。

7) 单击 Next(下一步)。此时在左窗格中将突出显示登录页面与 Network Authentication(网络身份验证)。

### 2. 步骤 2：登录-指定身份验证设置

如果网站需要网络身份验证，请执行以下操作：

1) 在网络验证区域中，选择一种身份验证方法，然后输入 Client Certificate(网络凭据)。

2) 要进行网络身份验证，选择使用客户端证书。

3) 在证书存储区中，选择下列其中一项，然后选择 My(我的)或 Root(根)单选按钮：

- 本地计算机(Local Machine)：在证书存储区域，WebInspect 使用用户选择的本地计算机上的证书。
- 当前用户(Current User)：在证书存储区域，WebInspect 对当前用户使用用户选择的证书。

4) 要查看证书信息区域的详细信息，先选择一个证书。

5) 单击 Next(下一步)，该应用程序身份验证页面出现。

6) 如果网站需要登录宏，请执行以下操作：

   a) 选择 Use a login macro for this site(为网站使用一个登录宏)。

   b) 在 Automated Login Sequence (Login Macro)(自动登录序列(登录宏))文本框中，如果想清除先前选定的宏，单击右端文本框中的 X。

   c) 单击 Edit(编辑)录制一个新登录宏，或单击 ... (浏览)按钮导航并选择编辑现有登录宏。

7) 单击 Next(下一步)。出现组织任务页面，在左窗格中的个人网站突出显示最佳设置。

### 3. 步骤 3：配置主动学习设置

查看推荐的扫描设置，来提高网站覆盖率，按照以下方式进行查看。

#### 1) 探查

该 WebInspect 探查对目标网站进行了初步项目，以确定某些设置是否应该被修改。如果修改是必需的，探查将返回一个建议列表，用户可以接受或拒绝该建议。

#### 2) 设置

a) 接受或拒绝该建议。如果要拒绝，清除相关的复选框。

b) 如有必要，提供所需的信息。

c) 单击 Next(下一步)。

#### 3) 信息

如果探查不建议修改，扫描向导将显示消息“没有设置更改建议，您目前的扫描设置最适合这个网站。”网站任务的增强覆盖突出显示在左窗格中。

#### 4) 增强覆盖网站

为增强覆盖应用程序，浏览到应用程序中的关键位置来提高覆盖率。

#### 5) Web 表单值

向导扫描记录所有用户输入网页表单的值，在这里用户可以查看和修改值，这是被保存在扫描设置的一部分。在工具栏中的 Web 窗体部分，可单击 Export(导出)将值保存到一个单独的文件，或者单击 Import(导入)使用现有的值集。扫描设置(包括 Web 表单的值)可以使用默认值，用户可在将来的扫描中修改。

单击 Next(下一步)。

最终审查页面和突出显示的配置详细选项在左窗格中出现。

### 4. 步骤 4：查看设置

要配置详细选项，指定以下任一设置。

#### 1) 重用识别误报

选择 False Positives(误报)框，重用 WebInspect 已确定的误报。

#### 2) 流量分析

a) 使用 Web 代理工具，选择 Launch and Direct Traffic(启动和直接流量)，通过 Web 代理服务器来使用 Web 代理工具，来检查通过 WebInspect 发出的 HTTP 请求和目标服务器返回的响应。

Web 代理是一个独立的代理服务器，用户可在桌面上配置和运行。Web 代理允许通过扫描仪、Web 浏览器或提交的 HTTP 请求和从服务器接收响应的任何其他工具监控流量。Web 代理是用于调试和渗透扫描的工具；当浏览一个网站时，用户可以查看每个请求和服务器响应。

b) 选择 Traffic Monitor(流量监视器)框来显示和审查 WebInspect 从服务器接收到的相关 HTTP 响应发送的每个 HTTP 请求。

在扫描一个网站时，WebInspect 只显示网站层次结构以及发现漏洞的那些会话。但如果选择 Enable Traffic Monitor(启用流量监视器)，WebInspect 允许显示和审查 WebInspect 从服务器接收到的相关 HTTP 响应发送的每个 HTTP 请求。

c) 单击 Next(下一步)。出现验证设置和开始扫描页面，在左窗格中突出显示配置详细选项。

d) 要保存扫描设置，选择 Click here to save settings(单击此处保存设置)。

e) 在立即扫描区域，检查扫描设置，然后单击 Start Scan(开始扫描)开始一个扫描。

### 5. 导入 HP 统一功能测试(UFT)文件

如果已安装 HP 统一功能测试应用程序，WebInspect 可以检测到它，并允许用户将一个 UTF 文件(.usr)导入到工作流扫描，以提高扫描的彻底性和攻击面。

导入一个 UTF(.usr)文件转换成 WebInspect 向导扫描：

1) 启动一个向导扫描，然后选择 Workflow Scan(工作流扫描)作为扫描类型。工作流扫描选项的附加文本会出现：

HP 统一功能测试已被侦破。可导入脚本来提高安全测试的彻底性。

2) 单击 Next(下一步)按钮。

3) 在认证部分，Application Authentication(应用程序身份验证)被自动选择。

4) 在管理工作流屏幕上，单击 Import(导入)。出现导入脚本对话框。在导入脚本对话框，可以：

- 键入文件名。
- 通过单击一个...找到用户的文件和.urs 扩展来浏览文件。从下拉文件类型选择 HP Unified Functional Testing(HP 统一功能测试)，然后导航到该文件。
- 单击编辑以启动 HP 统一功能来测试应用程序。

5) (可选)在导入脚本对话框，可以选择以下任一选项：

- 在导入期间显示 HP 统一功能测试 UI(Show HP Unified Functional Testing UI during import)
- 导入后打开脚本结果(Open script result after import)

6) 选择要导入的文件，然后单击 Import(导入)。在文件被成功导入后，该文件会出现在工作流表中。

7) 从工作流表执行下列操作之一：

- 记录(Record)：启动 WebInspect 统一宏录制。
- 编辑(Edit)：允许使用 WebInspect 统一的 Web 宏记录修改文件。
- 删除(Delete)：从工作流表中删除脚本。
- 导入(Import)：导入另一个文件。
- 导出(Export)：用 webmacro 格式指定名称和位置，并保存为一个文件。

8) 单击 Next(下一步)。

当第一个.usr 脚本文件添加到列表中，其名称(或默认名称)会出现在工作流程表并允许主机表被添加到窗格中。

添加另一个.usr 脚本文件可以添加更多的允许主机。已启用的任意主机对所有列出的工作流的.usr 脚本文件可用。不只是为它加入了工作流的.usr 文件。向导扫描将列出所有工作流文件，并要求列出所有的允许主机，以及其复选框是否被选中。如果选择了一个允许主机的复选框，WebInspect 将爬行或审计该主机的响应。如果复选框未选中，WebInspect 不会爬行或审计该主机的响应。

此外，如果一个特定的工作流.usr 脚本使用参数，当在列表中选择工作流，宏会显示一个宏参数表。根据需要编辑参数的值。

9) 完成变更或增加工作流表，在向导扫描的指导下继续完成设置并运行扫描。

### 7.3.2　基础扫描

当启动一个基础扫描时，会出现扫描向导窗口。

随后窗口中默认显示的选项都是 WebInspect 的默认设置。用户所做的任何更改都将只用

于此扫描。如果在窗口底部单击 Settings(Default)(设置(默认))访问 WebInspect 设置窗口，用户的任何操作都是暂时性更改。要更改默认设置，必须从 Edit(编辑)菜单中选择 Default Scan Settings(默认扫描设置)。

页面具体内容如图 7-45 所示。

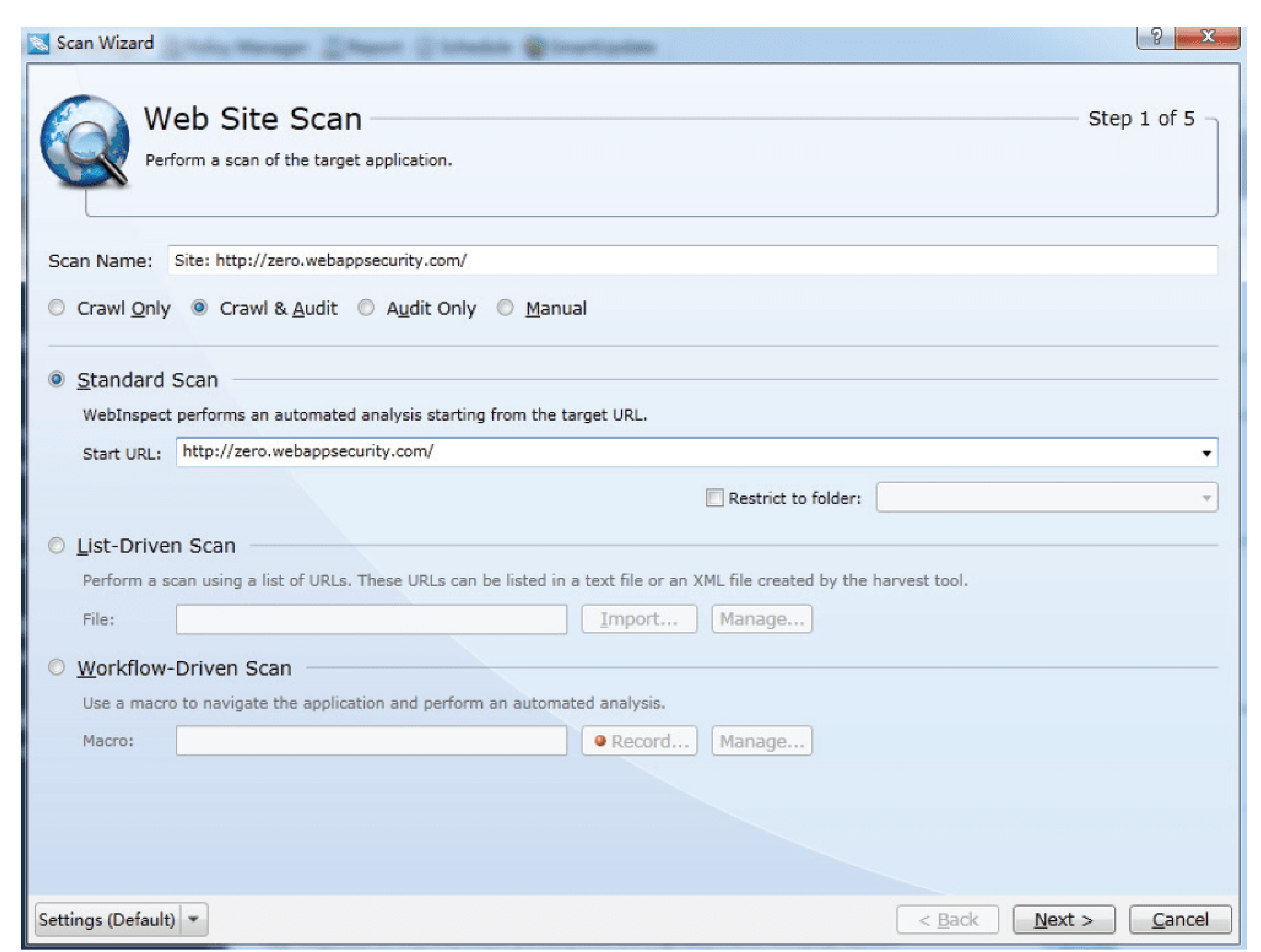



图 7-45　Scan Wizard(扫描向导)窗口

扫描名称(Scan Name)：给本次扫描设置一个名称。

选择扫描模式，包括以下四种：

仅爬行(Craw Only)：测试比较大型的网站，建议先爬行，再对内容审计。

爬行和审计(Crawl&Audit)：此选项为默认选项，该选项会以最快的速度完成爬行和审计。测试小规模的网站时，建议使用此选项。

仅审计(Audit Only)：对扫描的结果进行审计。

手动(Manual)：类似于被动检测，WebInspect 会模拟一个浏览器，然后根据用户的操作，自动扫描和审计用户访问的内容，所有操作都必须在 WebInspect 模拟出来的浏览器上进行。

选择扫描类型，包括以下三种：

标准扫描(Standard Scan，手动扫描)：用户需要提供待扫描网站的地址。勾选 Restrict to folder(限制到文件夹)复选项可以限制扫描的深度，如只扫描目录、扫描目录及其子目录、扫描目录及其父目录。

列表驱动扫描(List-Driven Scan)：用户需要提供一个待扫描 URL 的列表，文件格式为.txt 或者.xml。WebInspect 根据列表内容扫描和审计。

工作流驱动扫描(Workflow-Driven Scan)：录制一个工作流的宏，WebInspect 根据工作流

的操作，自动扫描和审计访问的内容。

任务 1：选择扫描选项

1) 在 Scan Name(扫描名称)文本框中，输入一个名称或扫描的简要说明。

2) 要载入之前保存的设置(可选)：

a) 单击 Settings(设置)按钮的下拉箭头。

b) 执行以下操作之一：

选择 From File(来自文件)，打开标准的文件选择窗口，然后选择一个文件，然后单击 Open(打开)。

从下拉列表中选择一个文件。

3) 选择以下扫描模式之一：

- 仅爬行(Crawl Only)：此选项会完全映射一个网站的分层数据结构。当爬行已经完成，可以单击审计，以评估应用程序的漏洞。
- 爬行与审计(Crawl and Audit)：WebInspect 映射网站的分层数据结构和审计每个资源(页)。根据选择的默认设置，每个资源被爬行或在整个网站被爬行后，审计就可以进行漏洞评估。
- 仅审计(Audit Only)：WebInspect 应用于选定策略来确定漏洞风险的方法，但不爬行网站。网站上无链接遵循或评估。
- 手动(Manual)：手动模式可手动导航到选择访问的应用程序的任何部分。它不会爬行整个网站，仅记录那些遇到的手动导航网站的资源信息。此功能是最常用的，通过一个 Web 表单登录页面进入网站或定义一个离散子集。一旦通过网站完成导航，用户可评估审计结果来记录网站的安全漏洞。

4) 选择以下扫描类型之一：

- 标准扫描(Standard Scan)：WebInspect 从目标 URL 开始进行自动分析。这是开始一个新扫描的最常用方式。
- 手动扫描(Manual Scan)：手动爬行(步模式)可手动导航到选择访问的应用程序的任何部分。此选项仅当选择手动扫描模式时可用。
- 列表驱动的扫描(List-Driven Scan)：使用由扫描列表执行的 URL 扫描。每个 URL 必须是完全限定的，并且必须包括协议(例如，http://或 https://)。可使用一个文本文件(格式为逗号分隔列表，或每行一个网址)，或使用由 Files To URLs 实用程序生成的 XML 文件。
- 工作流驱动扫描(Workflow-Driven Scan)：WebInspect 只审计包括在先前记录宏中和并没有在审计过程中遇到的超链接 URL，不需要注销签名。使用这种类型的宏通常专注于一个特定应用程序的一部分。用户可以选择多个宏。

5) 如果选择 Standard Scan(标准扫描)，请按以下说明操作：

a) 在 Start URL 框中，键入或选择要检查网站的完整 URL 或 IP 地址。

b) 如果选择 Restrict to folder(对文件夹限制)，可以限制扫描的范围，从下拉列表中选择区域。选项包括：

- 仅目录(Directory only)：WebInspect 仅将爬行和/或审计用户指定的 URL。例如，如果选择了此选项，并指定 URL 为 www.mycompany/one/two/，WebInspect 将仅评估

two 目录。

- 目录和子目录(Directory and subdirectories)：WebInspect 开始爬行和/或审计指定的URL，但不会访问目录树中任何较高的目录。
- 目录和父目录(Directory and parent directories)：WebInspect 开始爬行和/或审计指定的URL，但不会访问目录树中任何较低的目录。

6) 如果选择 Manual Scan(手动扫描)，输入 URL，如有必要，可选择 Restrict to folder(限制到文件夹)。参见标准扫描(上图)。

7) 如果选择 List-Driven Scan(列表驱动的扫描)，请执行下列操作之一：

- 单击 Import(导入)，然后选择一个文本文件或包含要扫描 URL 列表的 XML 文件。
- 单击 Manage(管理)创建或修改 URL 列表。

注：此功能也与 FilesToURLs 功能结合使用。

8) 如果选择 Workflow-Driven Scan(工作流驱动的扫描)，请执行下列操作之一：

- 单击 Manage(管理)，选择、编辑、录制、导入、导出或删除一个宏。
- 单击 Record(记录)并创建宏。

**注意**　*在一个扫描中可以包括多个宏。*

9) 单击 Next(下一步)。

任务 2：提供身份验证和连接信息

页面具体内容介绍如图 7-46 所示。

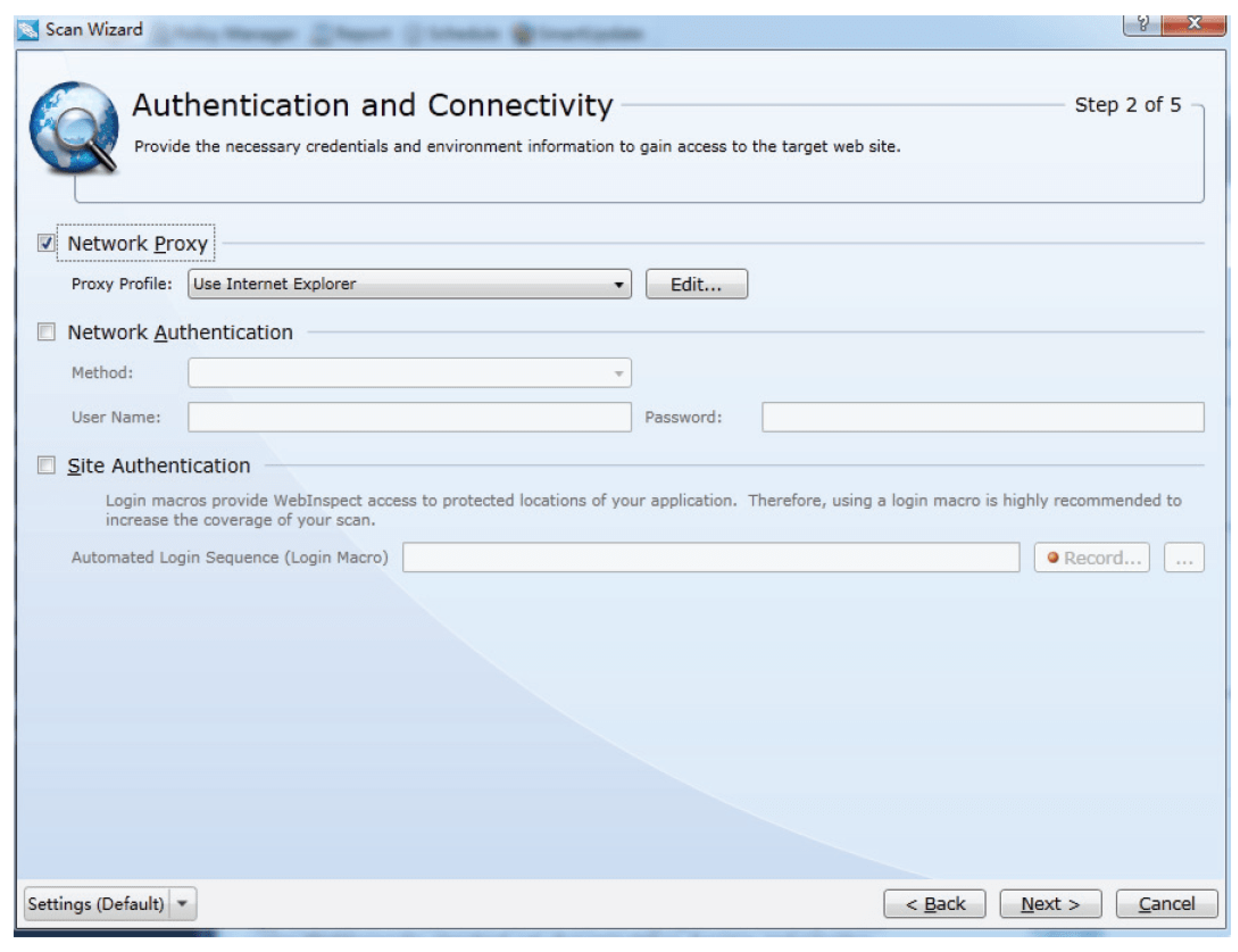



图 7-46　身份验证和连接信息

如果提供的 URL 不在许可证的范围内，WebInspect 会报错。

网络代理(Network Proxy)：默认使用 IE 的网络代理，也可以修改为其他代理。

网络身份验证(Network Authentication)：若网站的认证只是简单认证(如 Windows 认证)，选择该选项即可。

网站身份验证(Site Authentication)：如果网站需要认证，在这里选之前录制的登录宏，利用宏来进行登录。

1) 如果需要通过代理服务器访问目标网站，选择 Network Proxy(网络代理)，然后从 Proxy Profile(代理配置)文件列表中选择选项：

- 自动检测(Autodetect)：使用 WPAD 来定位一个代理自动配置文件，并用它来配置浏览器的 Web 代理服务器。
- 使用 Internet Explorer(Use Internet Explorer)：从 IE 导入代理服务器信息。
- 使用 PAC 文件(Use PAC File)：从 PAC 文件加载代理设置。如果选择此选项，单击 Edit(编辑)，然后进入 PAC 的位置(URL)。
- 使用显式代理设置(Use Explicit Proxy Settings)：指定代理服务器设置。如果选择此选项，单击 Edit(编辑)，然后进入代理信息。
- 使用 Mozilla Firefox 浏览器(Use Mozilla Firefox)：从 Firefox 导入代理服务器信息。

**注意**　选择使用浏览器的代理设置并不能保证通过代理服务器访问网络。如果 Firefox 浏览器的连接设置配置为“无代理”，或者不选中 Internet Explorer 设置“使用代理服务器为 LAN”，将不会使用代理服务器。

2) 选择 Site Authentication(网站身份验证)，如果需要服务器身份验证。那么选择一种身份验证方法，输入用户的网络凭据。

3) 选择网站验证使用包含用户名和密码的录制宏，让用户登录到目标网站。该宏还必须包含一个“退出条件”，这表明当一个意外退出发生时，WebInspect 可以重新运行该宏再次登录。

- 单击 ... 选择一个宏。如果选择一个宏后，希望使用 Web 宏记录修改宏，单击 Edit(编辑)。
- 单击 Record... 创建一个宏。

4) 单击 Next(下一步)。

任务 3：指定覆盖范围和彻底性

页面具体内容介绍如图 7-47 所示：

框架(Framework)：如果系统构建在一些常用框架之上，HP WebInspect 可以针对这些框架或者组件进行扫描优化，但仍需提供登录宏等必备组件，目前只支持 Oracle ADF Faces。Oracle ADF Faces 包含 100 多个 JSF 组件，用户可以使用它们为 Java EE 应用程序构建更丰富的用户界面。Oracle ADF Faces 还包含当今 JSF 开发人员最需要的许多框架特性。

扫描范围(Crawl Coverage)：默认为最高级别，级别越高，扫描范围越广，速度越慢；级别越低，扫描范围越小，速度越快。这里需要平衡扫描范围和扫描时间。

校验策略(Audit Depth)：选择一个检验策略，我们常用的策略是 OWASP Top 10 2010。

WebInspect 会自动进行测评，优化之后的扫描，以便更快更好地执行操作。

1) 为对一个应用程序进行优化设置，使用 Oracle 应用开发框架 Faces 组件或 IBM WebSphere Portal，选择 Framework(框架)，然后从 Optimize scan for(优化扫描)列表中选择 Oracle ADF Faces 或 WebSphere Portal。惠普可能开发其他覆盖设置，并使其智能更新。

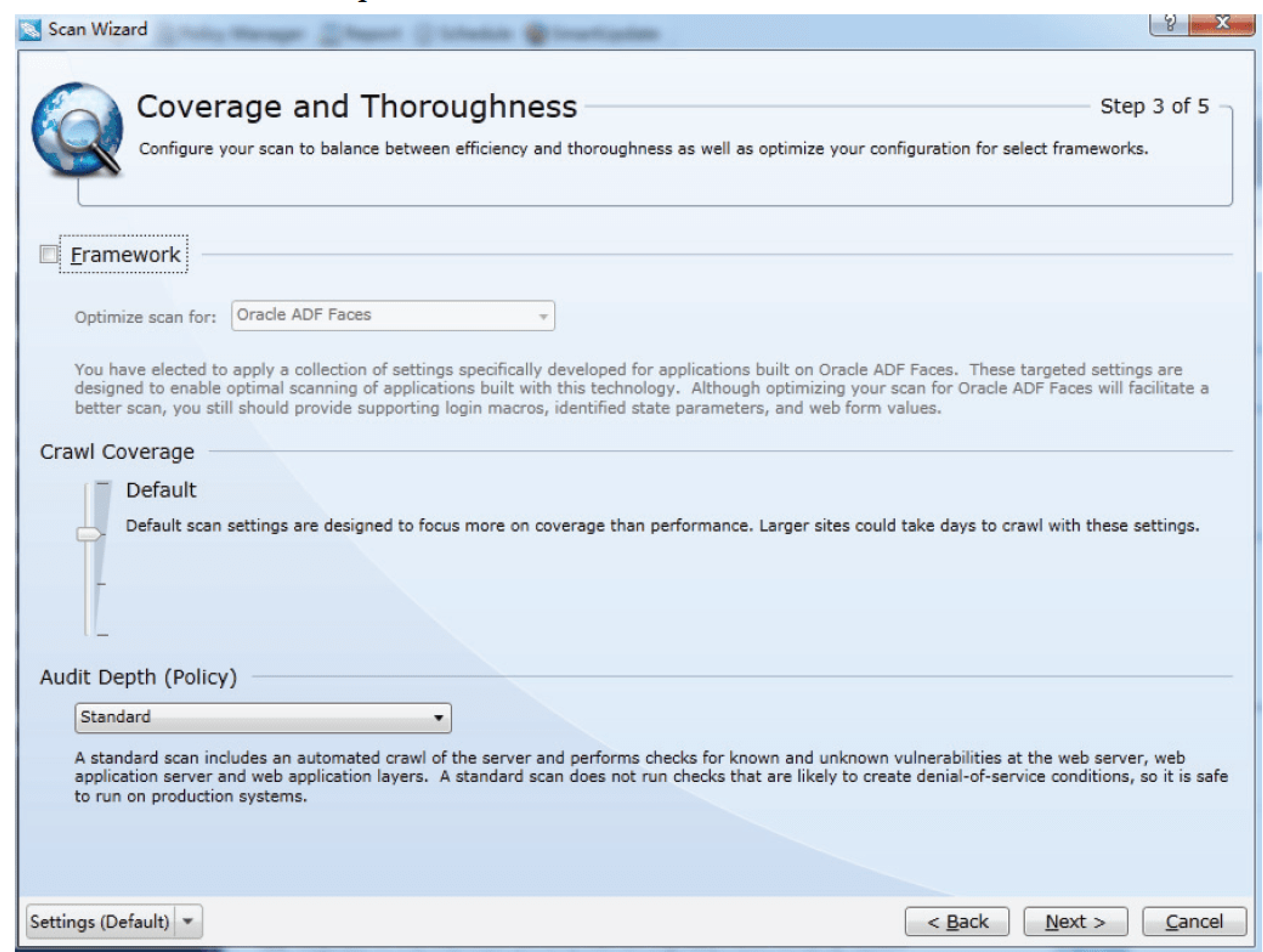



图 7-47　指定覆盖范围和彻底性

2) 使用 Crawl Coverage(爬行覆盖)滑块指定爬行设置。

此滑块是否会被启用，取决于选择的扫描模式。与此滑块相关的选项卡也取决于用户的选择。如果启用，滑块将允许选择四种爬行位置之一。各位置表示设定值的特定集合，由下面的选项卡表示：

a) 彻底(Thorough)

“彻底”使用以下设置：

- 冗余页检测：关闭
- 最大单链接点击数：20
- 最大 Web 表单提交：7
- 创建脚本事件会话：开
- 每页最大脚本事件：2000
- 每个会话允许的动态表单数：无限

注意　动态表单是在执行脚本时被修改的 HTML 表单。如果需要很长时间来审计一个脚本密集部位，将每个会话动态表单的数量限制到零次或一次来缩短审计时间。

b) 默认(Default)

默认使用以下设置：

- 命中次数包括参数：真
- 冗余页检测：关闭
- 最大单链接点击数：5
- 最大 Web 表单提交：3
- 创建脚本事件会话：关
- 每页最大脚本事件：1000
- 每个会话允许的动态表单数：无限

c) 正常(Normal)

正常使用以下设置：

- 命中次数包括参数：真
- 冗余页检测：关闭
- 最大单链接点击数：5
- 最大 Web 表单提交：2
- 创建脚本事件会话：关
- 每页最大脚本事件：300
- 每个会话允许的动态表单数：1
- 命中次数包括参数：假

d) 快速(Quick)

快速使用以下设置：

- 冗余页检测：开
- 最大单链接点击数：3
- 最大 Web 表单提交：1
- 创建脚本事件会话：关
- 每页最大脚本事件：100
- 每个会话允许的动态表单数：0
- 命中次数包括参数：假

如果单击 Settings(设置)(打开高级设置窗口)，并更改与四个滑块位置中的一个相冲突的设置，滑块创建第五位置标记 Customized Coverage Settings(自定义覆盖设置)。

3) 从 Audit Depth (Policy)(审计深度(策略))列表中选择策略。

这个列表是否会被启用，这取决于步骤 1 中选择的扫描模式。

4) 单击 Next(下一步)。

任务 4：提供详细的扫描配置信息，如图 7-48 所示。

如果单击 Profile(配置)文件(或者当出现该页面时选择 Run Profiler Automatically(自动运行分析器))，WebInspect 对目标网站进行初步审查，以确定某些设置是否要修改。如果要修改，分析器返回一个建议列表，用户可以接受或拒绝。

启动分析器每次访问这个页面，会选择自动运行分析器。要手动启动分析器，单击 Profile(配置)文件，在 Settings(设置)部分显示结果。

1) 接受或拒绝该建议。要拒绝，清除相关的复选框。

2) 如有必要，提供所需的信息。

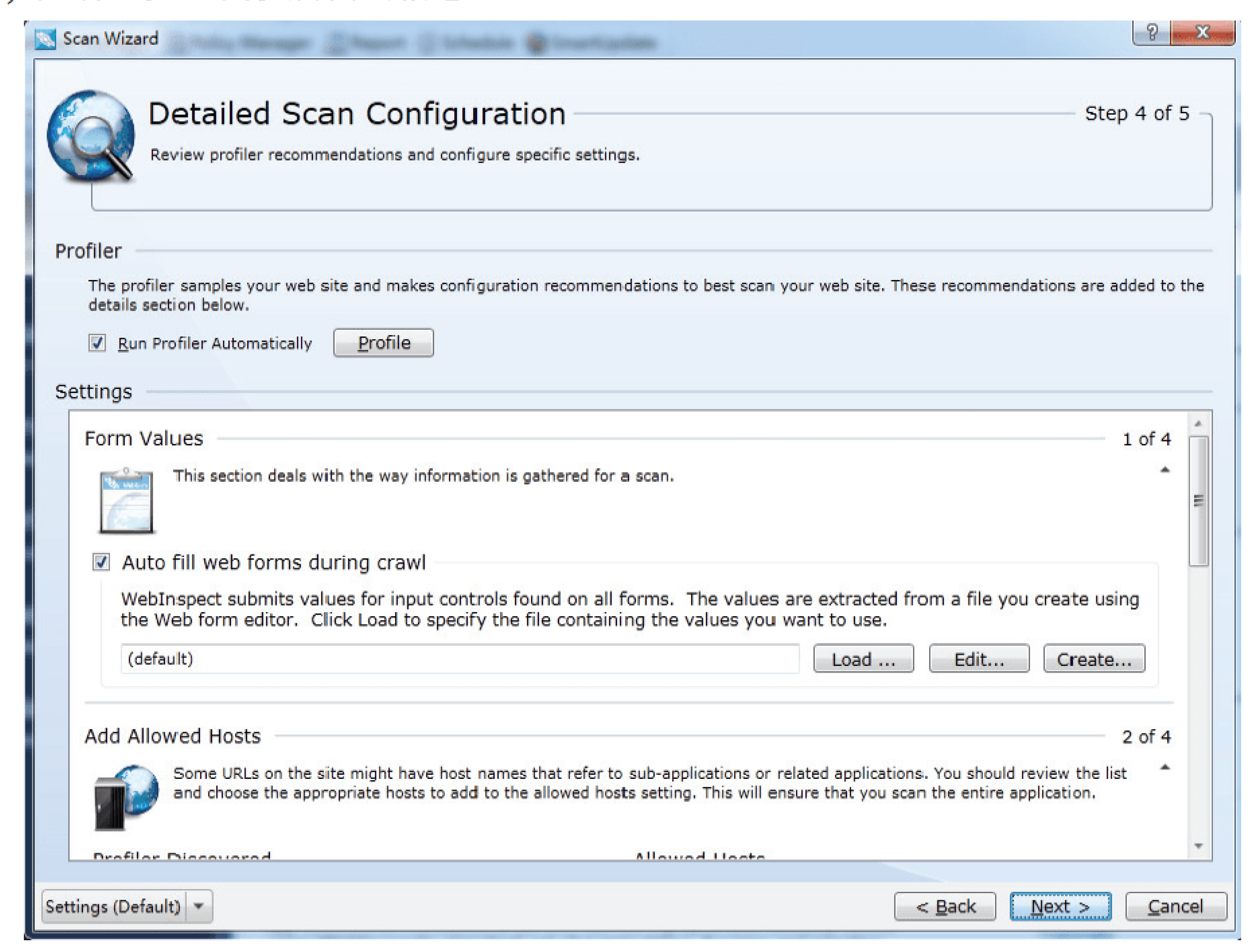



图 7-48　详细的扫描配置

3) 单击 Next(下一步)。

任务 5：启动或计划扫描，如图 7-49 所示。

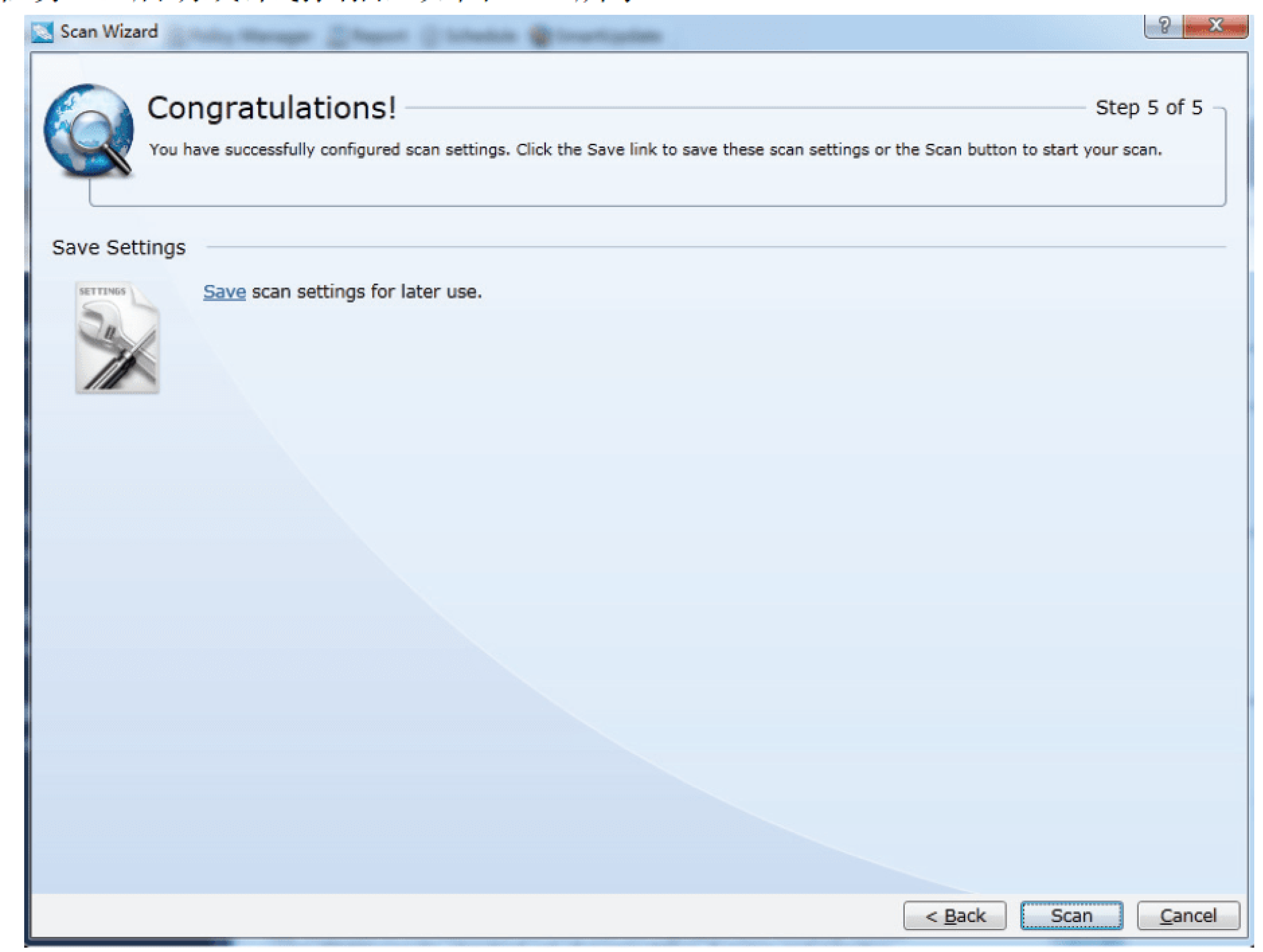



图 7-49　保存并启动扫描

此窗口中的内容有所不同，这取决于用户的选择和配置(与计划扫描类似，不详细介绍)。

1) 上传到 AMP/ WebInspect Enterprise 扫描模板

当连接到企业服务器(AMP 或 WebInspect Enterprise)时，用户可将此扫描设置发送到企业服务器，这将创建一个扫描模板。但必须分配给一个角色(在 AMP 或 WebInspect Enterprise)。单击上传超文本的链接。

2) 保存设置

用户可保存此扫描设置，这将允许以后再次使用此扫描设置。单击保存超文本的链接。保存扫描设置，单击 Scan(扫描)按钮开始扫描。

任务 6：网站测试

1) WebInspect 的工作界面如图 7-50 所示。

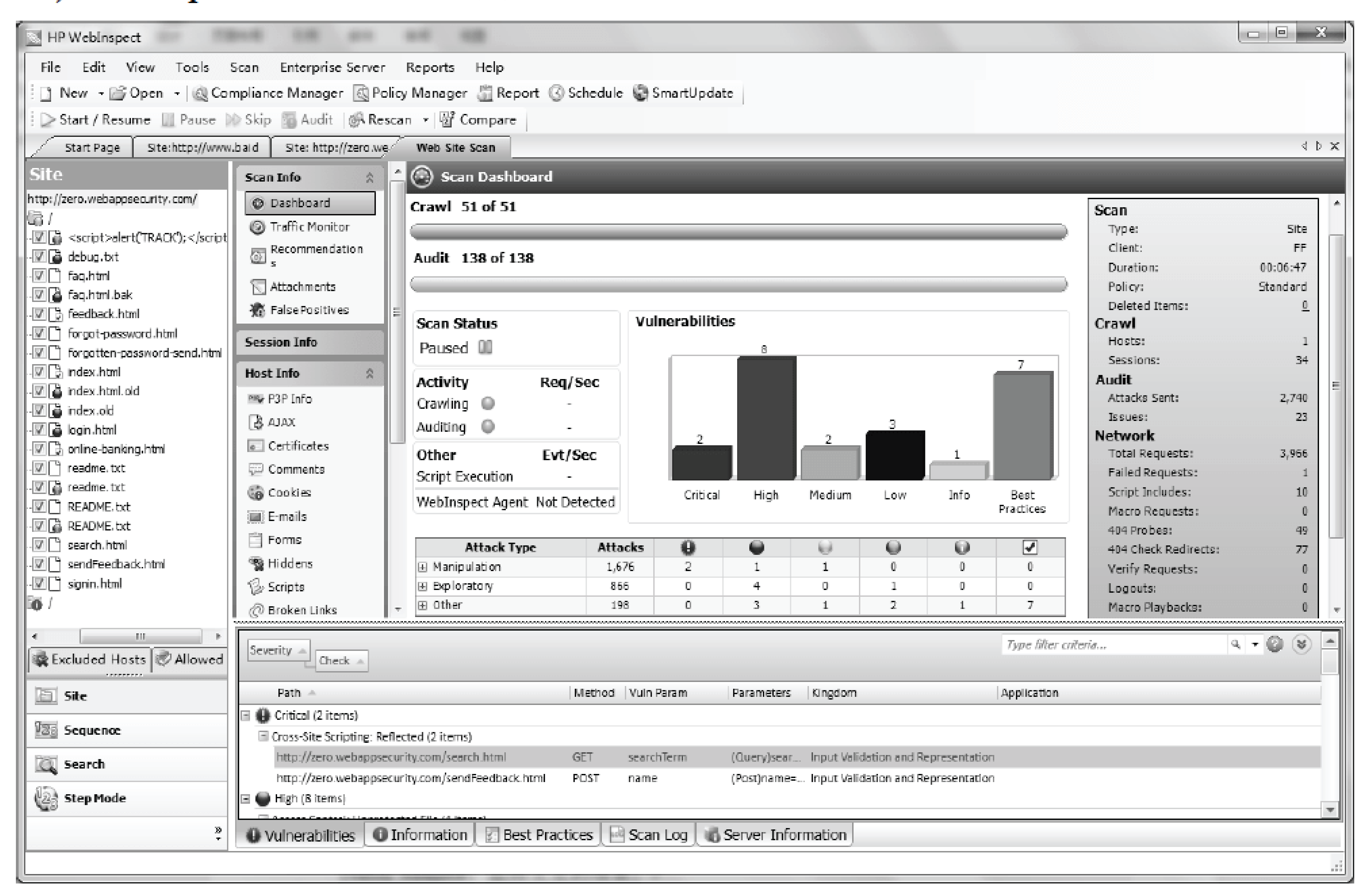



图 7-50　WebInspect 的工作界面

重要信息说明：

Site：列出扫描出的网站内容 JSP、Servlet、JS 等，根据网站树来排列。

Scan Info：查看扫描的总体信息。

Dashboard：以柱状图来显示扫描到的漏洞。

Traffic Monitor：监控交互的信息。

Session Info：查看一个具体漏洞的详细信息。

Vulnerability：该漏洞的详细介绍。

Web Browser：可在浏览器中查看发现的漏洞。

HTTP Request：查看 WebInspect 发送的 HTTP Request 的信息。

HTTP Response：查看待测试网站返回的 HTTP Response。

Details：查看 HTTP Request 头、HTTP Response 头、Session 信息。

Steps：查看 WebInspect 发现这个漏洞的详细执行步骤。

Host Info：查看待测网站的信息。

Vulnerability List：以列表的形式列出已扫描到的漏洞、WebInspect 的日志信息、针对这个漏洞的最佳实践等。在这里可以更改漏洞的级别，利用 WebInspect 自带的小工具来重现该漏洞，也可将该漏洞发送到 HP Quality Center 或者 IBM Rational Clear 中(如果已与其集成)。

2) 查看漏洞信息

这里是该漏洞的所有信息的汇总，在漏洞列表中右键单击要查看的漏洞，在右键快捷菜单中选择 Review Vulnerability(审阅漏洞)，打开的界面如图 7-51 所示。

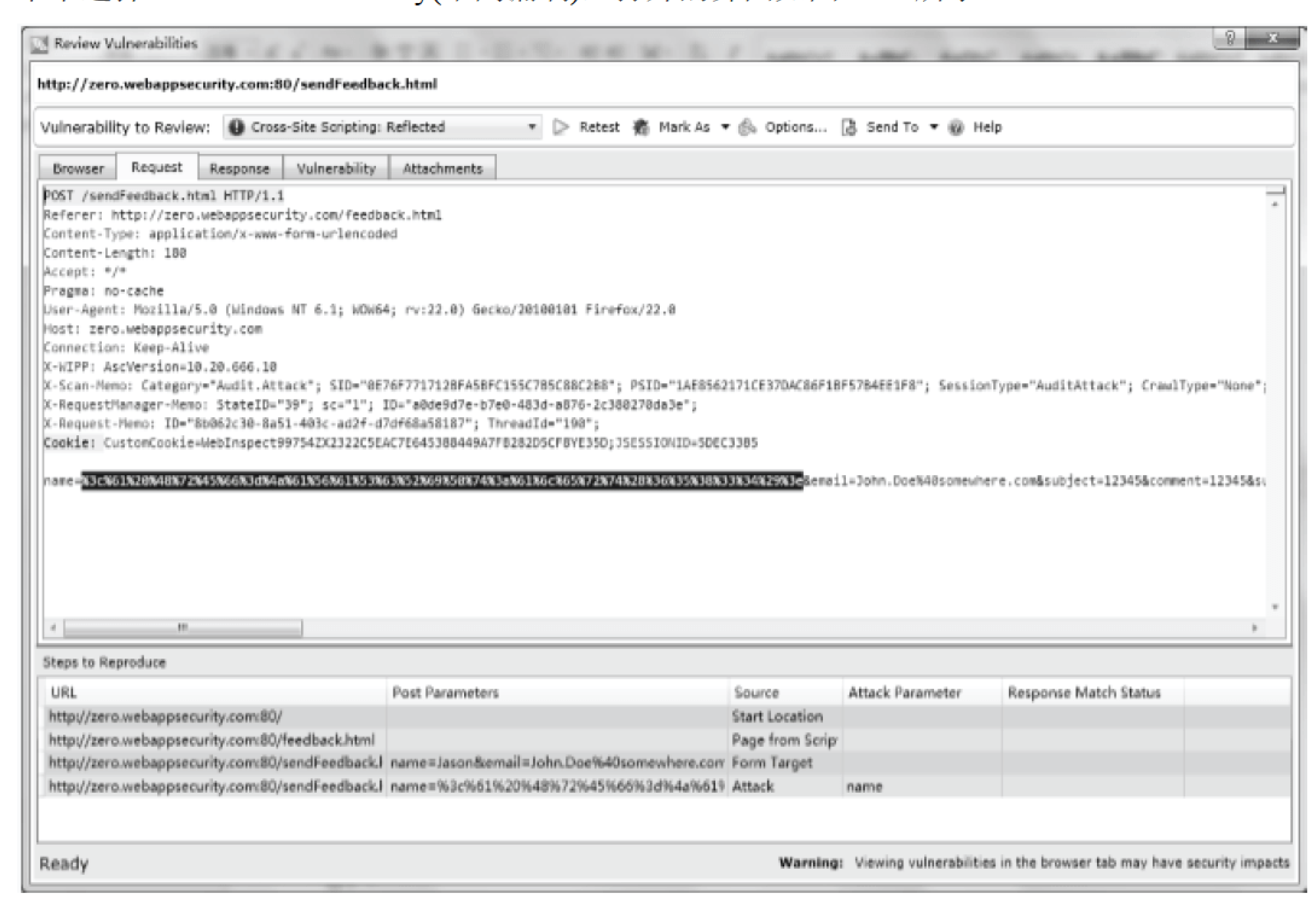



图 7-51　查看漏洞信息

Browser：在浏览器中查看该漏洞。

Request：WebInspect 发送的 HTTP Request 信息。

Response：待测网站返回的 HTTP Response 信息。

Vulnerability：查看该漏洞的详细信息介绍。

Attachments：查看该漏洞所添加的附件。

界面下方显示了重现步骤。

3) 利用 WebInspect 自带小工具重现漏洞

在漏洞列表中右键单击要重现的漏洞，在右键快捷菜单中选择 Tools(工具)，WebInspect 会高亮显示可用的小工具，这里以 HTTP Editor 为例进行介绍，如图 7-52 所示。

HTTP Editor 工作界面的上方是 WebInspect 发送的 HTTP Request 的详细信息，单击 Send(发送)，待测网站返回的 HTTP Response 信息会显示在工作界面的下方，选择 Browser(浏览)标签可在浏览器中查看该漏洞。

4) 更改缺陷级别

在漏洞列表中右键单击要更改缺陷级别的漏洞，在右键快捷菜单中选择 Change Severity(更改严重性)。

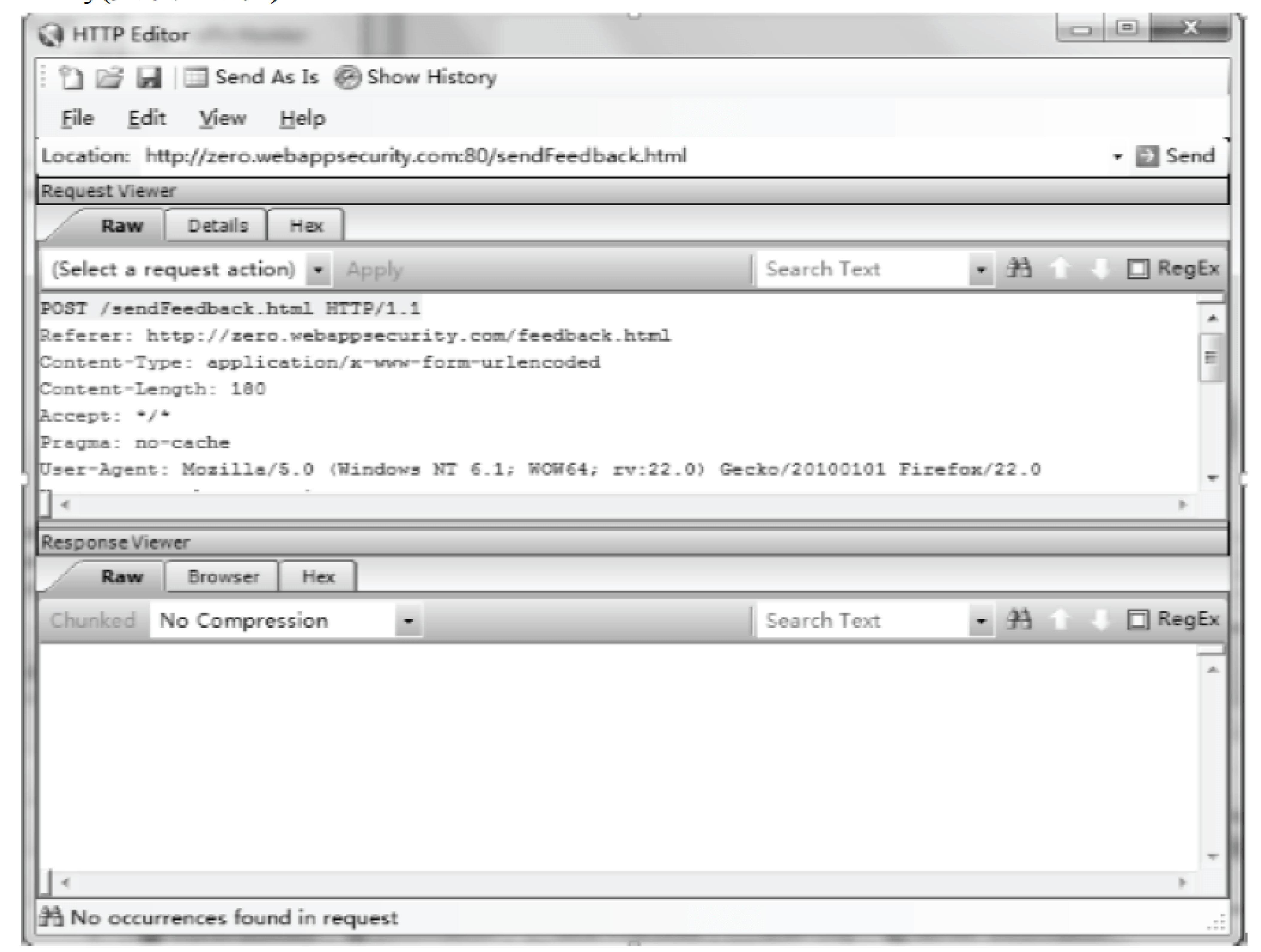



图 7-52　使用 HTTP Editor 重现漏洞

5) 在下次扫描中忽略该漏洞

用户可能发现有些漏洞是误报的，希望在下次扫描中忽略这些漏洞，那么可以在漏洞列表中右键单击要忽略的漏洞，在右键快捷菜单中选择 Mark As(标记为)，再选择 Ignored(忽略)或 False Positive(误报)。

6) 导出缺陷信息

在漏洞列表中右键单击要导出缺陷信息的漏洞，在右键快捷菜单中选择 Export(导出)，再选择 Selected item(s) to CSV(选中条目导出为 CSV)或 All items to CSV(全部条目导出为 CSV)，导出格式为.csv。

7) 将漏洞发送到 HP Quality Center

在漏洞列表中右键单击要发送的漏洞，如果 HP WebInspect 和 HP Quality Center 已经和 WebInspect 进行集成，在右键快捷菜单中选择 Send(发送)，再选择 HP Quality Center，然后填写漏洞信息，该漏洞会作为一个缺陷(bug)，发送到 HP Ouality Center 并根据 HP Quality Center 中定义的流程来处理。

任务 7：生成报告可以利用 WebInspect 自带的 Report Designer 来设计报告的格式，具体操作为选择 Reports(报告)，再选择 Report Designer(报告设计器)，打开 Report Designer。之后选择 File(文件)，再选择 New(新建)打开报告定义窗口，如图 7-53 所示。

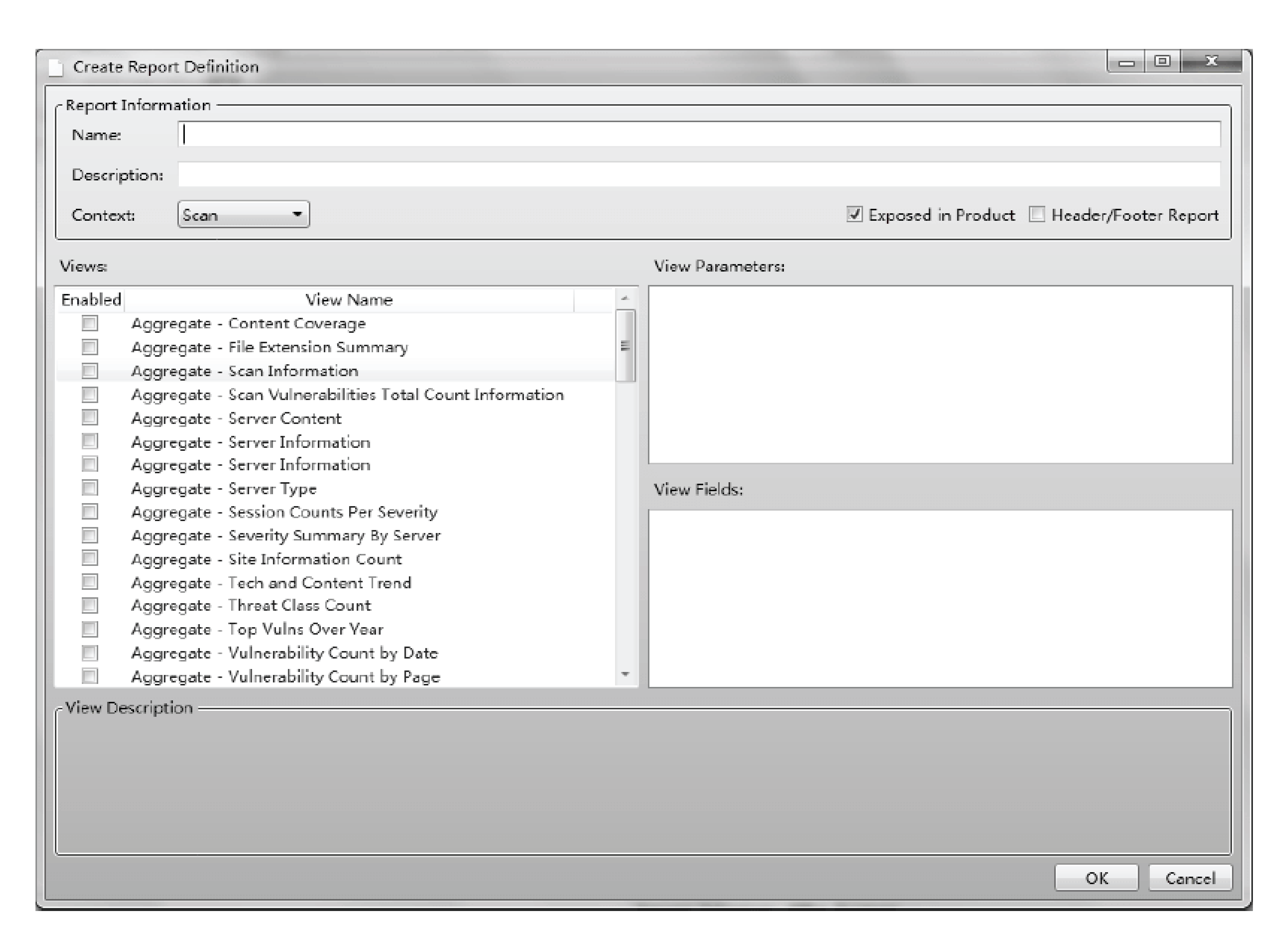



图 7-53　Create Report Defintion 窗口

Name：给报告设置一个名称。

Description：报告的具体描述。

Context：选择整个扫描的报告或者单个漏洞的报告。

Views：选择需要展示的视图。

View Parameters：该视图对应的参数。

View Fields：该视图对应的域。

View Description：视图的具体介绍。

单击 OK(确定)按钮，打开具体的格式设计界面，如图 7-54 所示。

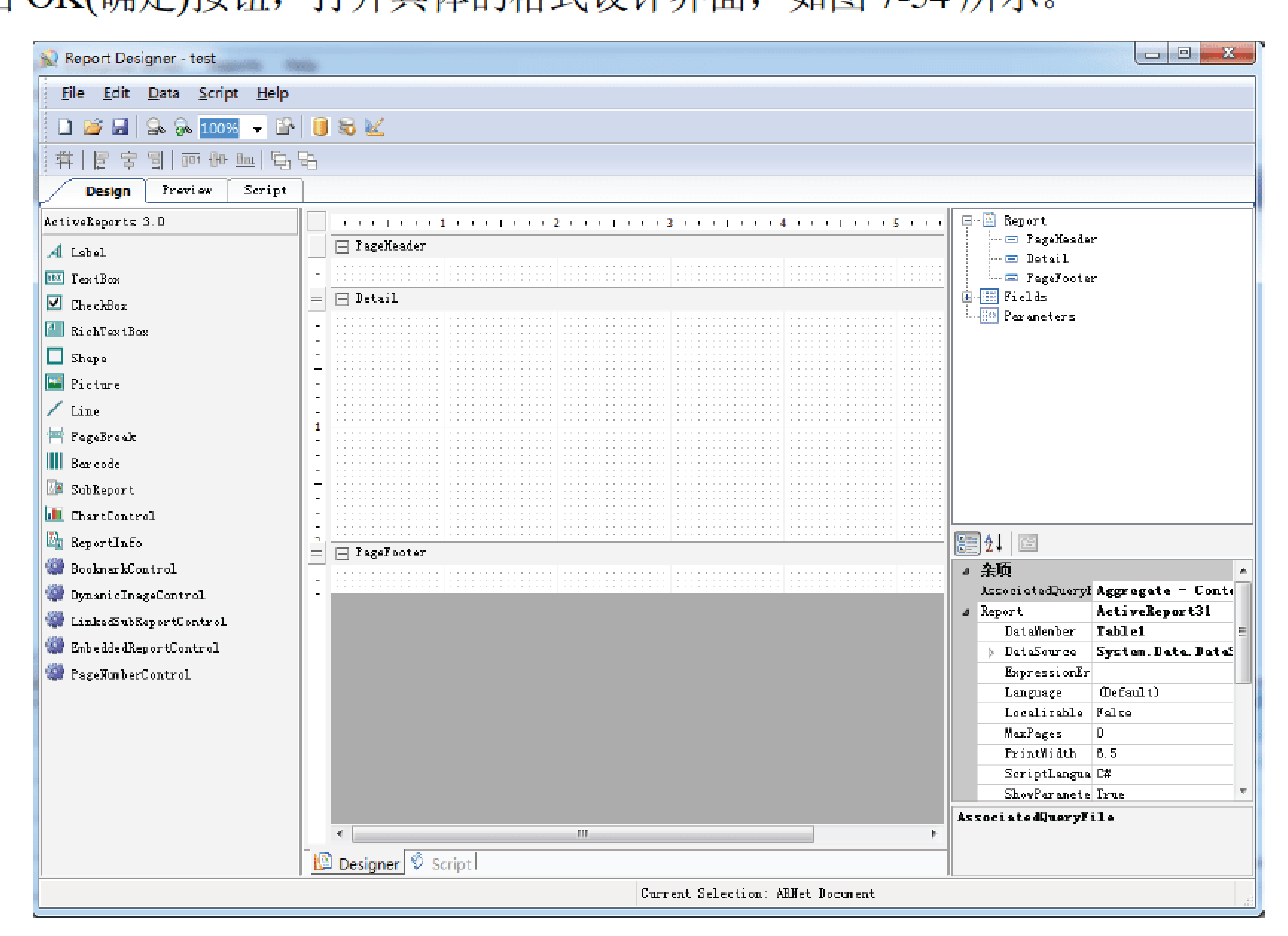



图 7-54　报告格式设计界面

设计好之后保存。

单击 Reports(报告)的下拉列表选择 Generate Reports(生成报告)或单击工作界面左侧的 Generate a Report(生成报告)，打开的界面如图 7-55 所示。

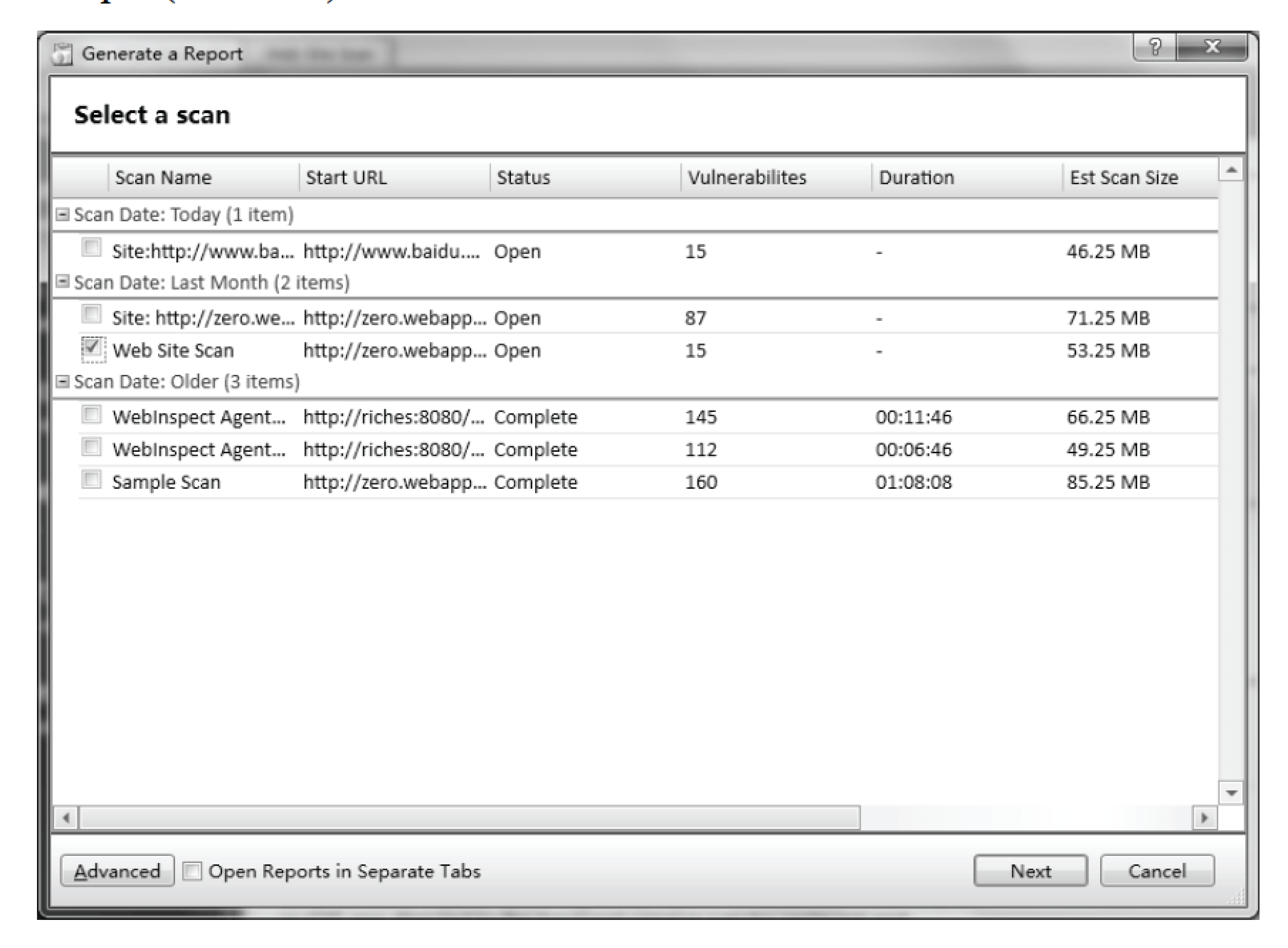



图 7-55　选择生成报告的扫描

选中要生成报告的扫描名称，单击 Next(下一步)，如图 7-56 所示。

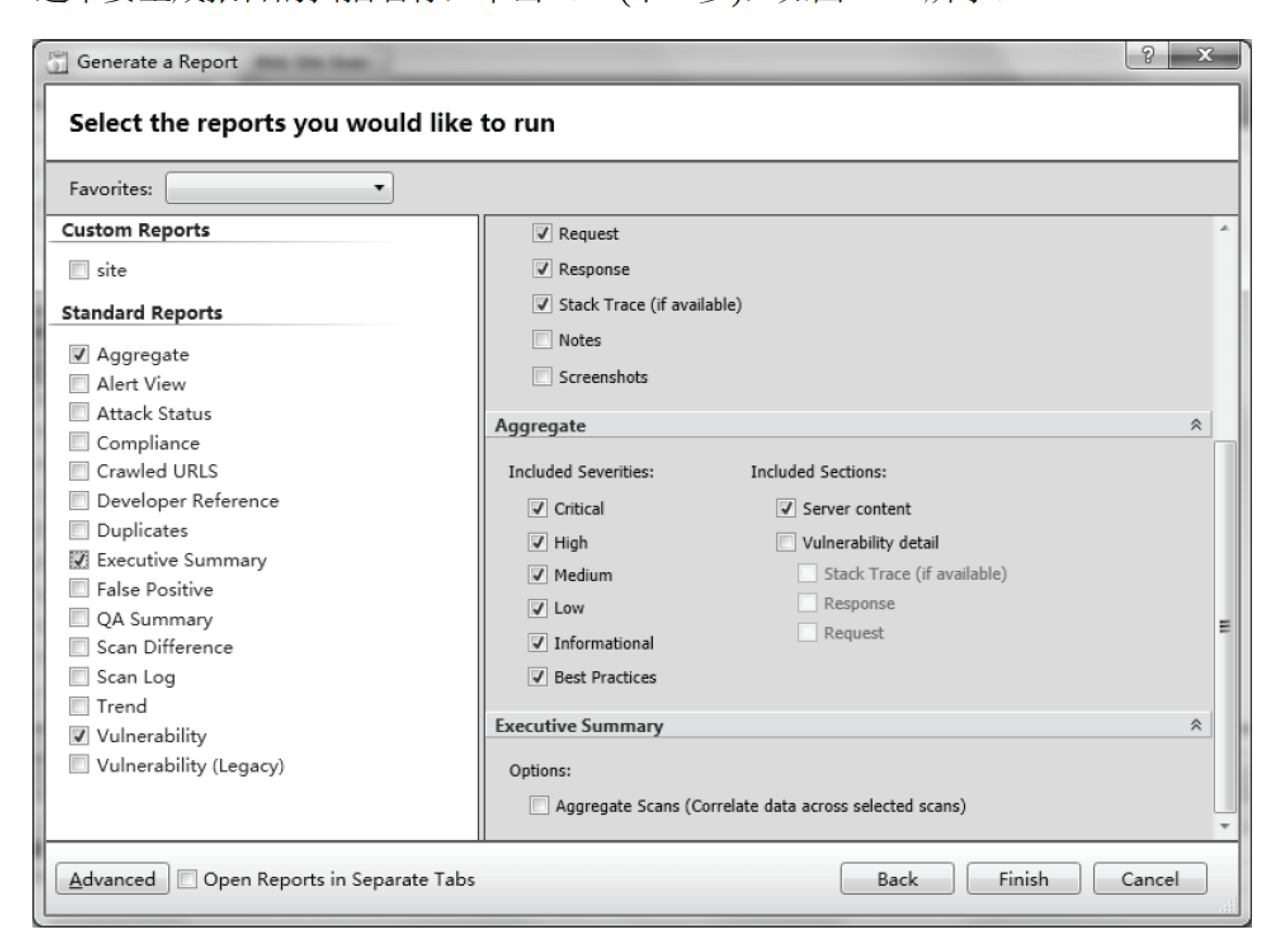



图 7-56　选择生成报告的内容

选择需要展示的信息内容，常用的有 3 个：Compliance、Executive Summary、Vulnerability。注意图 7-56 不要勾选太多选项，勾选太多会导致生成报告页数太多，生成速度也会很慢。单击 Finish(完成)按钮，WebInspect 会生成 PDF 格式的报告，如图 7-57 所示。

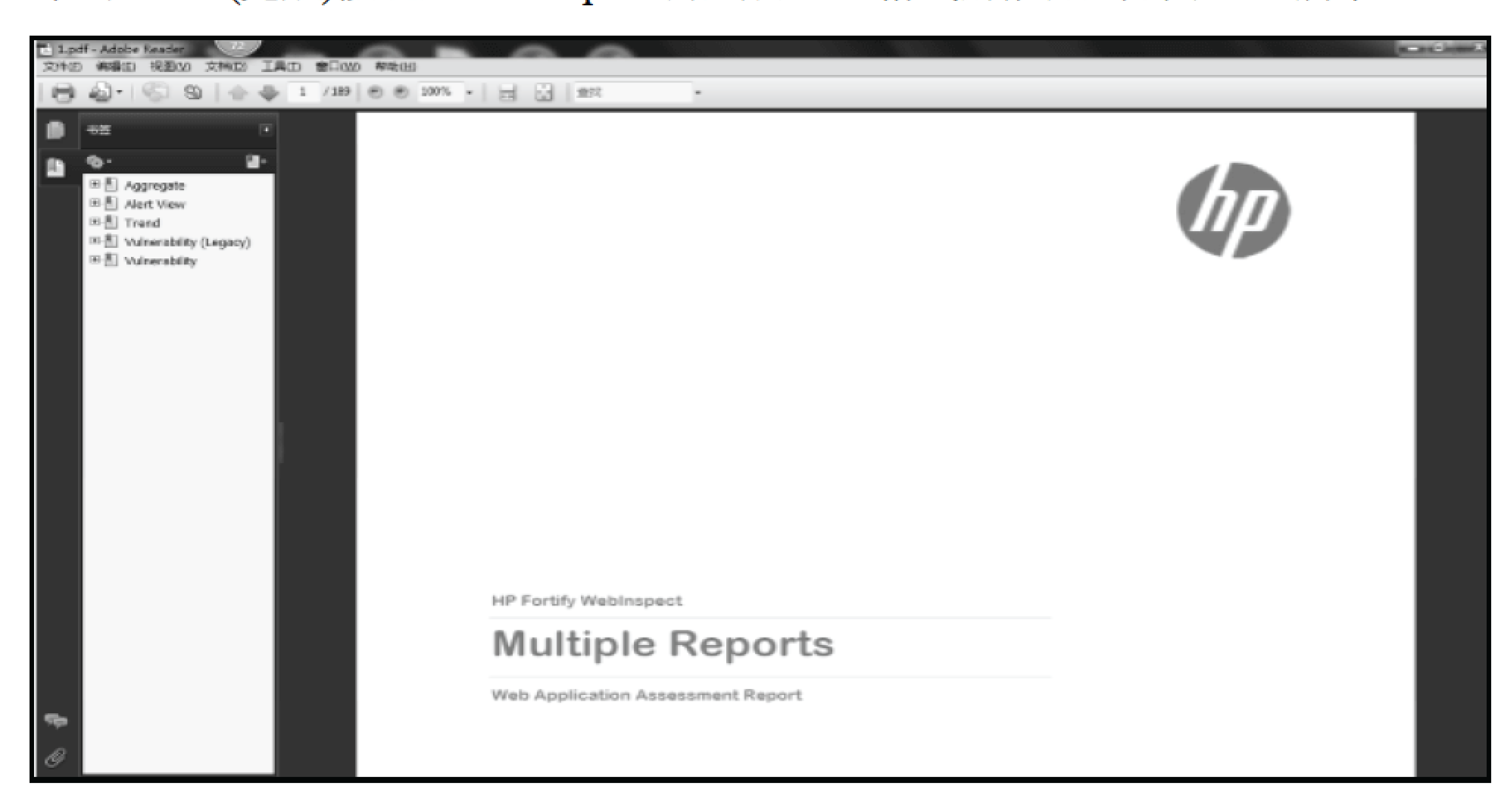
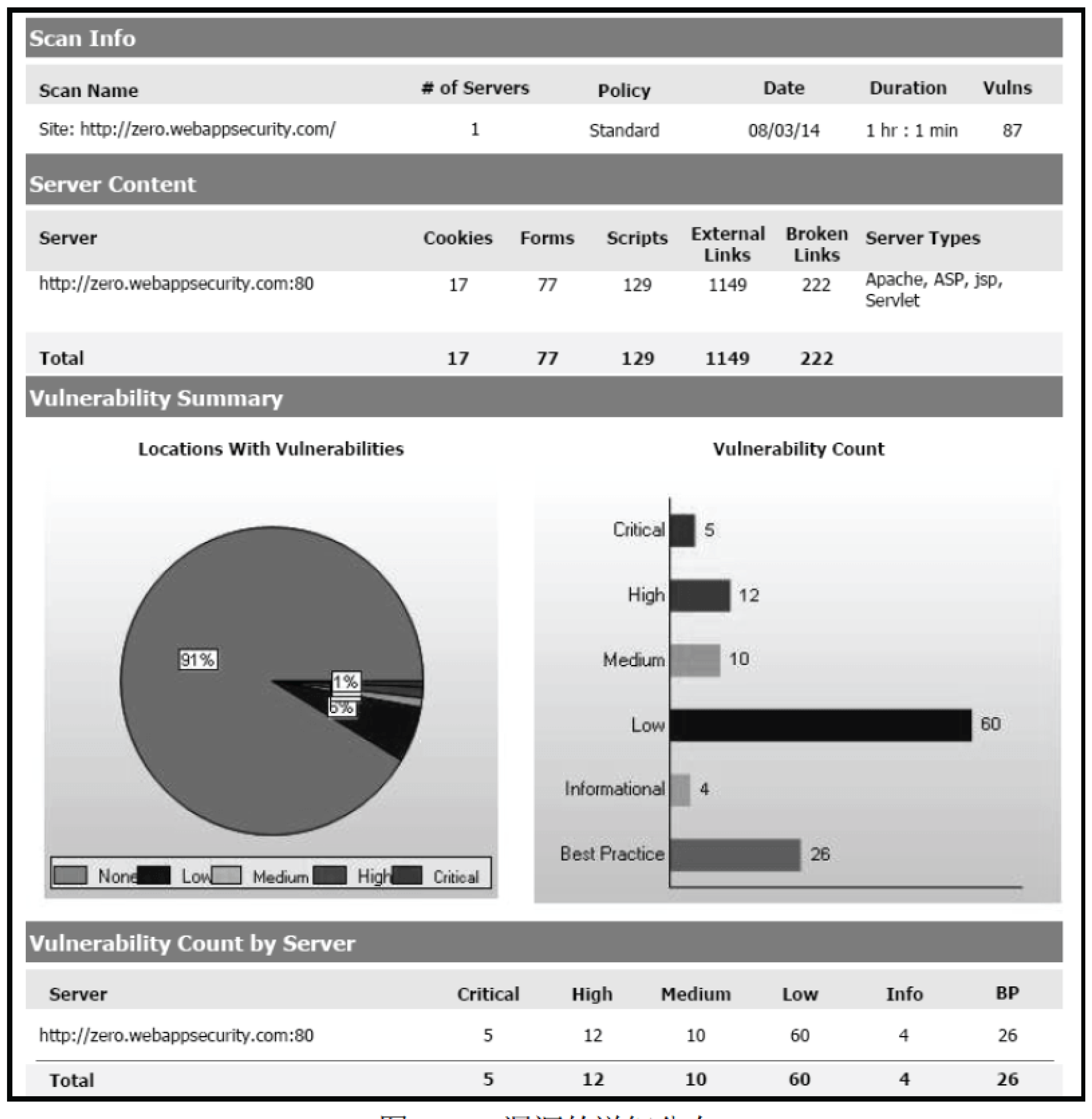



图7-57　生成PDF格式的报告

单击保存按钮后，可将生成的 WebInspect 报告保存到本地。报告中有详细的漏洞分布图，如图 7-58 所示。以及漏洞代码，如图 7-59 所示，显示为红色的代码是漏洞代码。



**Scan Info**

| Scan Name | # of Servers | Policy | Date | Duration | Vulns |
|---|---|---|---|---|---|
| Site: http://zero.webappsecurity.com/ | 1 | Standard | 08/03/14 | 1 hr : 1 min | 87 |

**Server Content**

| Server | Cookies | Forms | Scripts | External Links | Broken Links | Server Types |
|---|---|---|---|---|---|---|
| http://zero.webappsecurity.com:80 | 17 | 77 | 129 | 1149 | 222 | Apache, ASP, jsp, Servlet |
| Total | 17 | 77 | 129 | 1149 | 222 | |

**Vulnerability Summary**



**Vulnerability Count by Server**

| Server | Critical | High | Medium | Low | Info | BP |
|---|---|---|---|---|---|---|
| http://zero.webappsecurity.com:80 | 5 | 12 | 10 | 60 | 4 | 26 |
| Total | 5 | 12 | 10 | 60 | 4 | 26 |

图 7-58　漏洞的详细分布

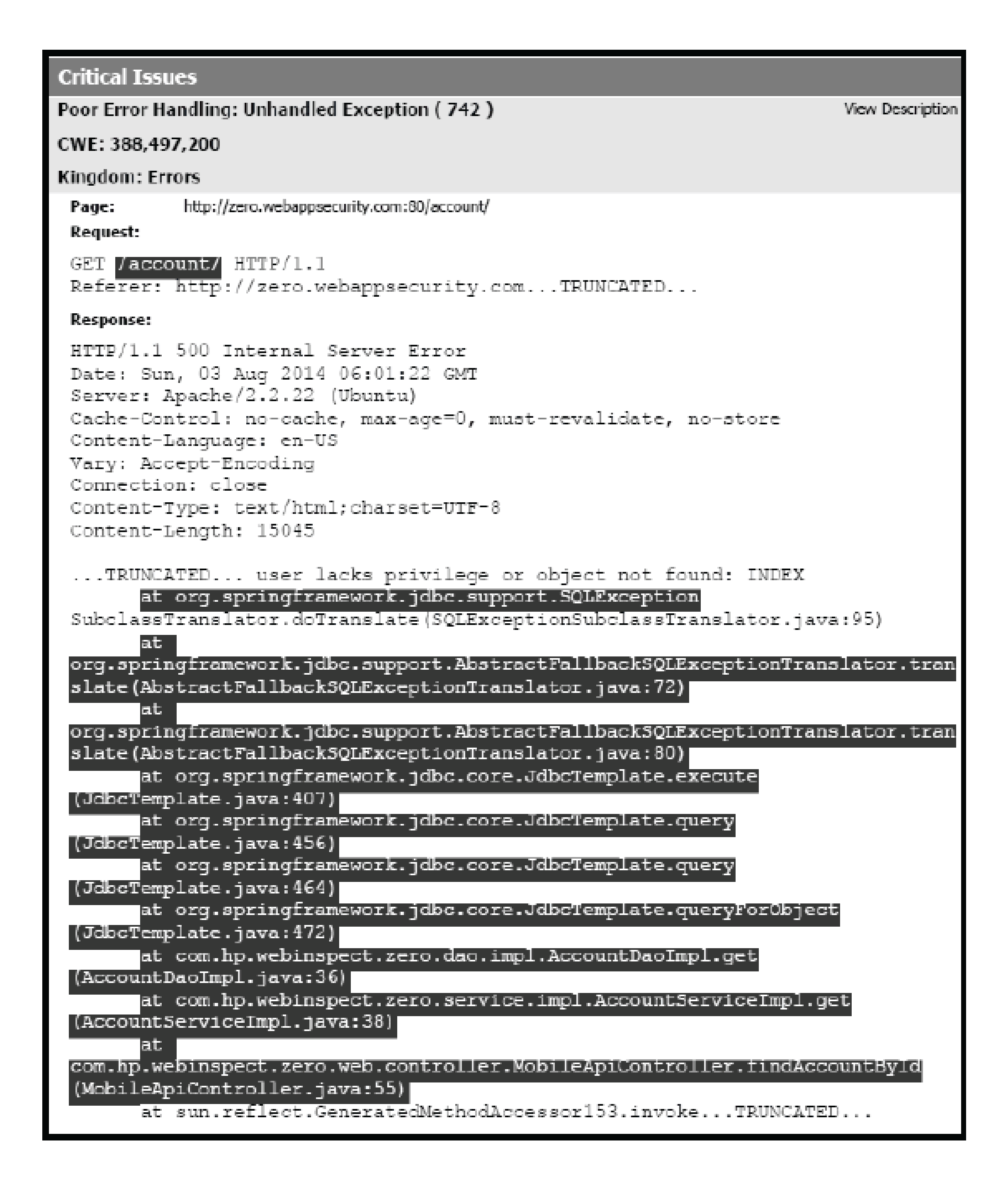



图 7-59　标红代码为漏洞代码

## 7.3.3　Web 服务扫描

当执行 Web 服务评估时，WebInspect 可以爬行 WSDL 网站，并提交每次操作中发现的每个参数的任意枚举值。这些值摘录自用户必须创建使用的 Web 服务设计器文件。然后在试图检测漏洞时，通过攻击每个参数去审计网站，如 SQL 注入。

Web 服务扫描界面如图 7-60 所示。

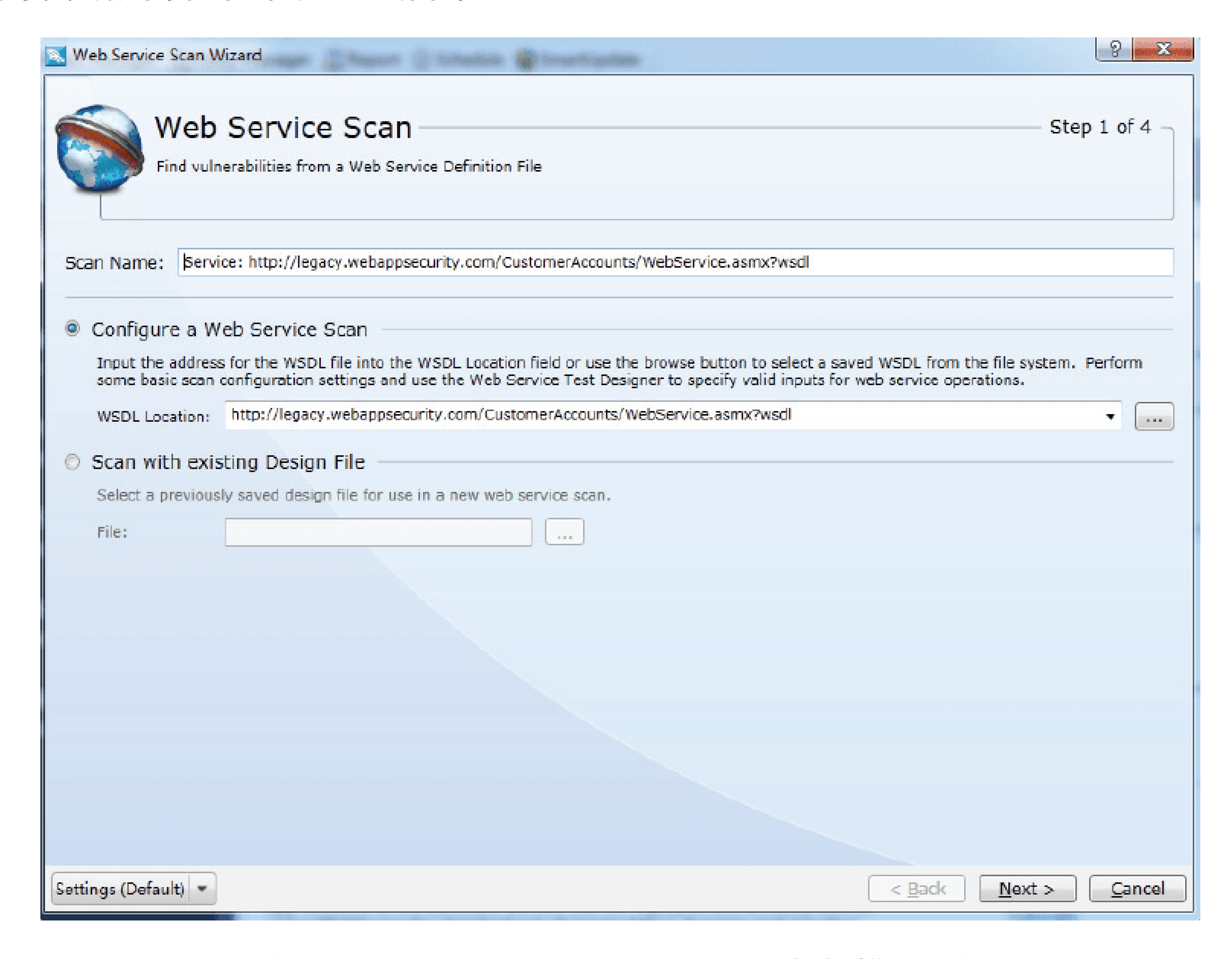



图 7-60　Web Service Scan(Web 服务扫描)界面

任务 1：选择一个 WSDL 或设计文件

1) 在扫描名称文本框中输入 Scan Name(扫描名称)。

2) 选择下列选项之一：

- 配置 Web 服务扫描(Configure a Web Service Scan)：输入或选择一个 WSDL 文件的完整路径和名称，或者单击[...]打开一个标准的文件选择对话框，选择一个 WSDL 文件。导入 WSDL 文件，随后运行 Web 服务测试设计配置包含在服务中的每个操作文件的值。
- 扫描现有设计文件(Scan with Existing Design File)：单击[...]打开一个标准的文件选择对话框，然后选择使用的 Web 服务测试设计文件，此文件由先前的 Web 服务测试设计(WSD)创建。此文件包含服务中每个操作的值。

3) 单击 Next(下一步)。

注意，WSDL 文件为接口文件，测试人员可向开发人员索要这个文件。

任务 2：提供身份验证和连接信息

1) 如果需要通过代理服务器访问目标网站，选择网络代理服务器，然后从代理配置文件列表中选择选项：

- 自动检测(Autodetect)：使用 Web 代理自动发现协议(WPAD)来定位代理自动配置文件，并使用它来配置浏览器的 Web 代理服务器设置。
- 使用 Internet Explorer(Use Internet Explorer)：从 IE 导入代理服务器信息。
- 使用 PAC 文件(Use PAC File)：从代理自动配置(PAC)文件加载代理设置。如果选择此选项，单击 Edit(编辑)，然后进入 PAC 的位置(URL)。
- 使用显式代理设置(Use Explicit Proxy Settings)：指定代理服务器设置。如果选择此选项，单击 Edit(编辑)，然后进入代理信息。
- 使用 Mozilla Firefox 浏览器(Use Mozilla Firefox)：从 Firefox 导入代理服务器信息。

注意，选择使用浏览器的代理设置并不能保证通过代理服务器访问网络。如果 Firefox 浏览器的连接设置配置为“无代理”，或者不选中 Internet Explorer 设置“使用代理服务器为 LAN”，那么代理服务器不会被使用。

2) 选择网络身份验证，如果需要服务器身份验证，则选择一种身份验证方法，输入网络凭据。

提供身份验证和连接信息界面，如图 7-61 所示。

3) 单击 Next(下一步)。

任务 3：详细扫描配置

1) 要创建一个设计测试文件，一个消息将提示用户启动 Web 服务测试设计。直到使用设计器创建一个 Web 服务设计(WSD)文件，扫描向导才会前进，如图 7-62 所示。

2) 如果已经选择了设计测试文件，用户可单击 Design(设计)，打开 Web 服务测试设计和编辑 WSD 文件，该文件包含应在扫描过程中提交的 WSDL 文件的值。

3) (可选)可选择以下选项：

- 通过 Web 代理服务器并启动 Direct Traffic。注意，正在进行计划扫描时，此选项不可用。
- 启用流量监控器。

4) 单击 Next(下一步)。

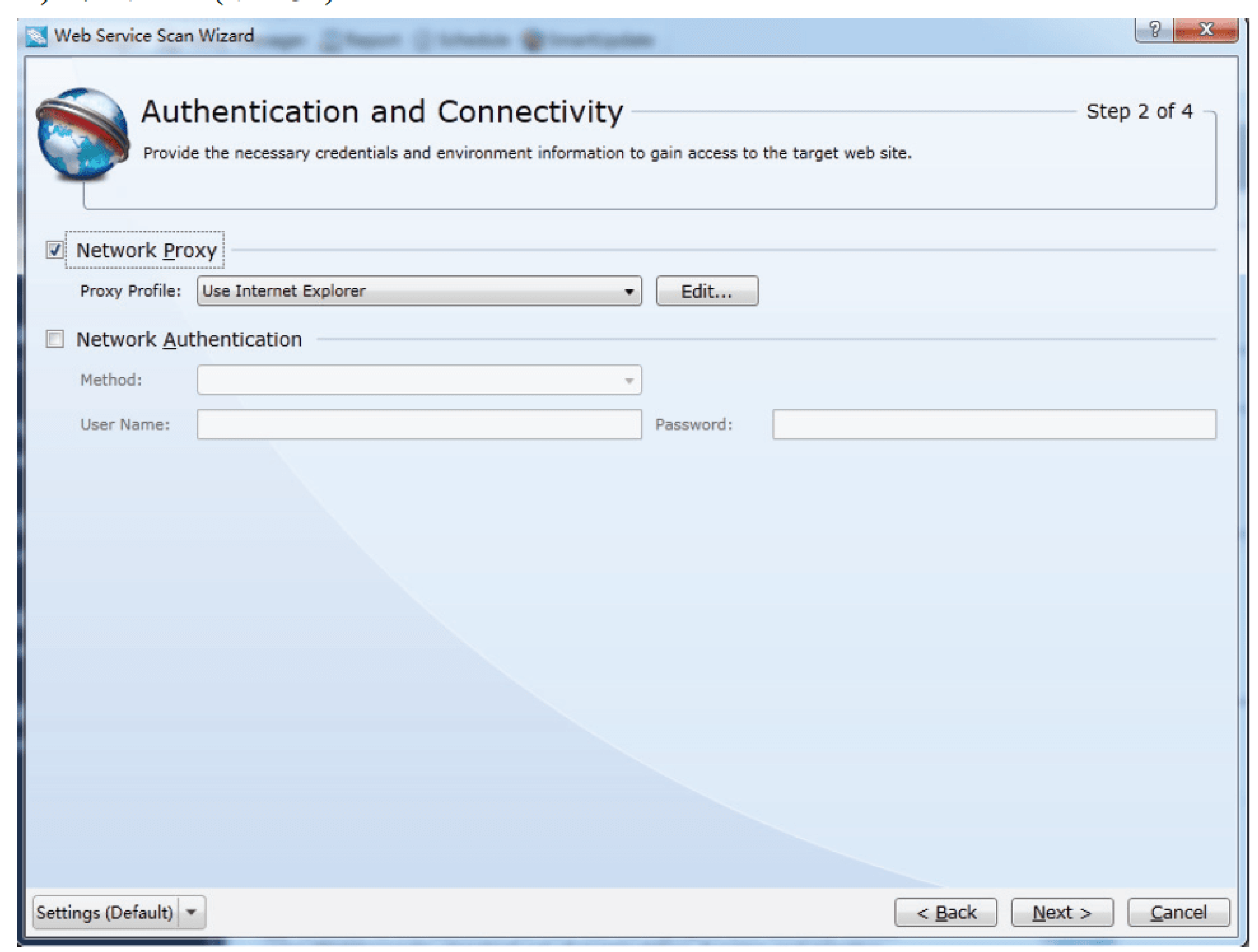



图 7-61　身份验证和连接信息界面

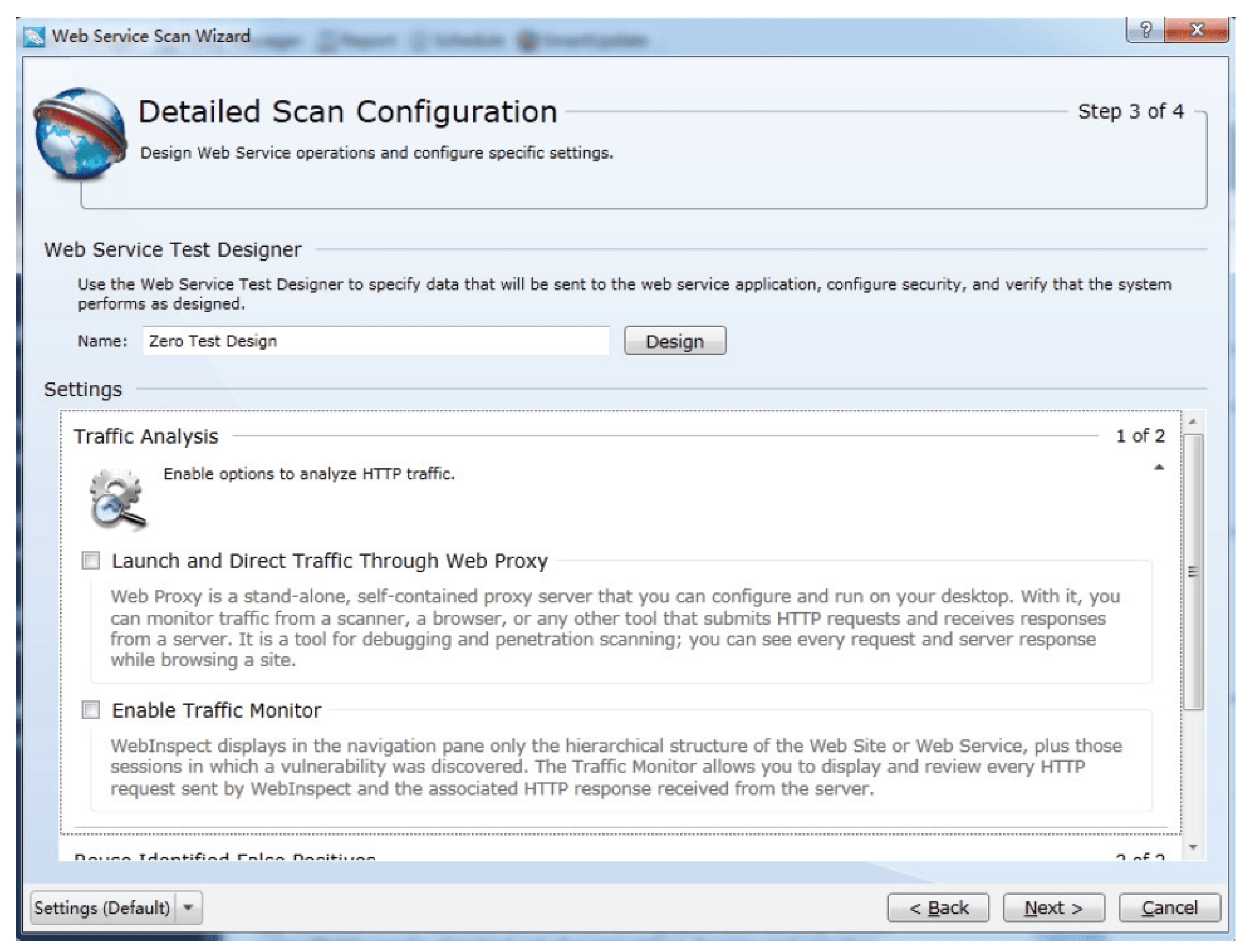



图 7-62　扫描详细配置

任务 4：启动或计划扫描(同基础扫描)。

### 7.3.4　企业扫描

企业扫描允许从企业网络的角度总览用户网络。WebInspect 自动发现所有可用端口的 IP 地址范围。然后，可选择使用哪些服务器来评估所有被发现的漏洞。

当启动一个企业扫描时，出现企业扫描窗口，如图 7-63 所示。Recurrence Pattern(循环模式)栏选择开始扫描的时间，可以选择立即开始或指定在每天的某个时间扫描。

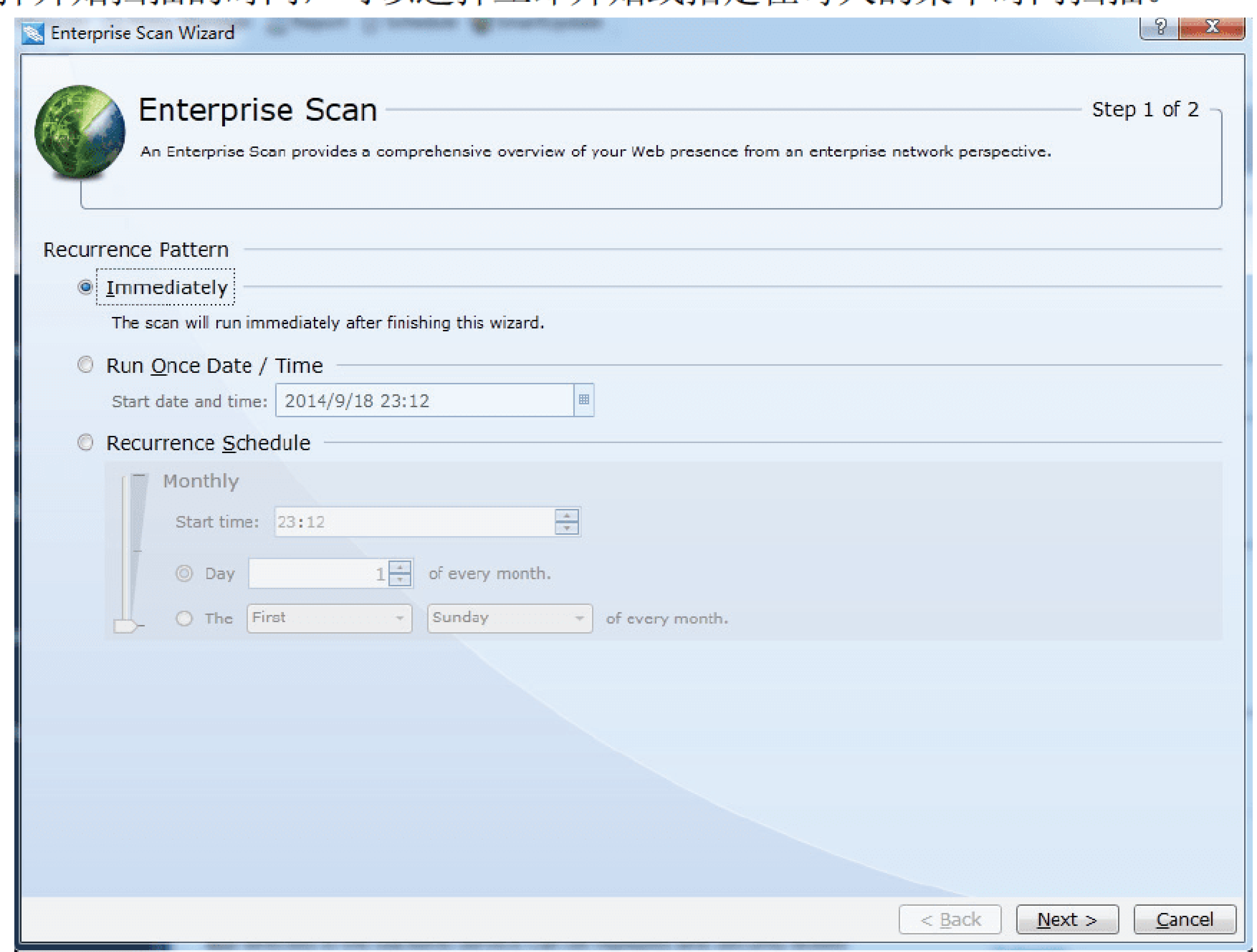



图 7-63　在企业扫描窗口中，可指定扫描时间

1) 进行扫描指定。选项包括：

- 立即(Immediately)。
- 运行一次(Run Once)：设置扫描开始时的日期和时间。可以单击下拉箭头以显示日历，选择日期。
- 循环计划(Recurrence Schedule)：使用滑块来选择一个频率(每日、每周或每月)。然后指定扫描应开始的时间(每周或每月)并提供其他计划信息。

单击 Next(下一步)继续进行设置，页面具体内容介绍如图 7-64 所示。

扫描名称(Enterprise Scan Name)：为本次扫描设置一个名称。

Web 发现(Web Discovery)：单击 Discover(发现)按钮，提供一个 IP 地址或一个 IP 地址的范围，WebInspect 会自动发现目标主机，被发现的主机会列在下方的主机列表内。

主机列表(Hosts to Scan)：可手动添加一台待测主机，也可以从外部导入。

2) 在 Enterprise Scan Name(企业扫描名称)框中，输入本次企业扫描的唯一名称。

从企业网络的角度来看，企业扫描允许总览网络存在。用户可以：

- 通过对指定范围内的 IP 地址和端口进行扫描发现所有可用的服务器。
- 输入单个 URL 或 IP 地址。
- 导入服务器列表(使用先前创建的列表)。

Web 查看

按照下面的步骤来查看 Web 服务器。

1) 单击 Discover(发现)。

Search For Web Servers (Web 服务器搜索)窗口会出现，如图 7-65 所示。

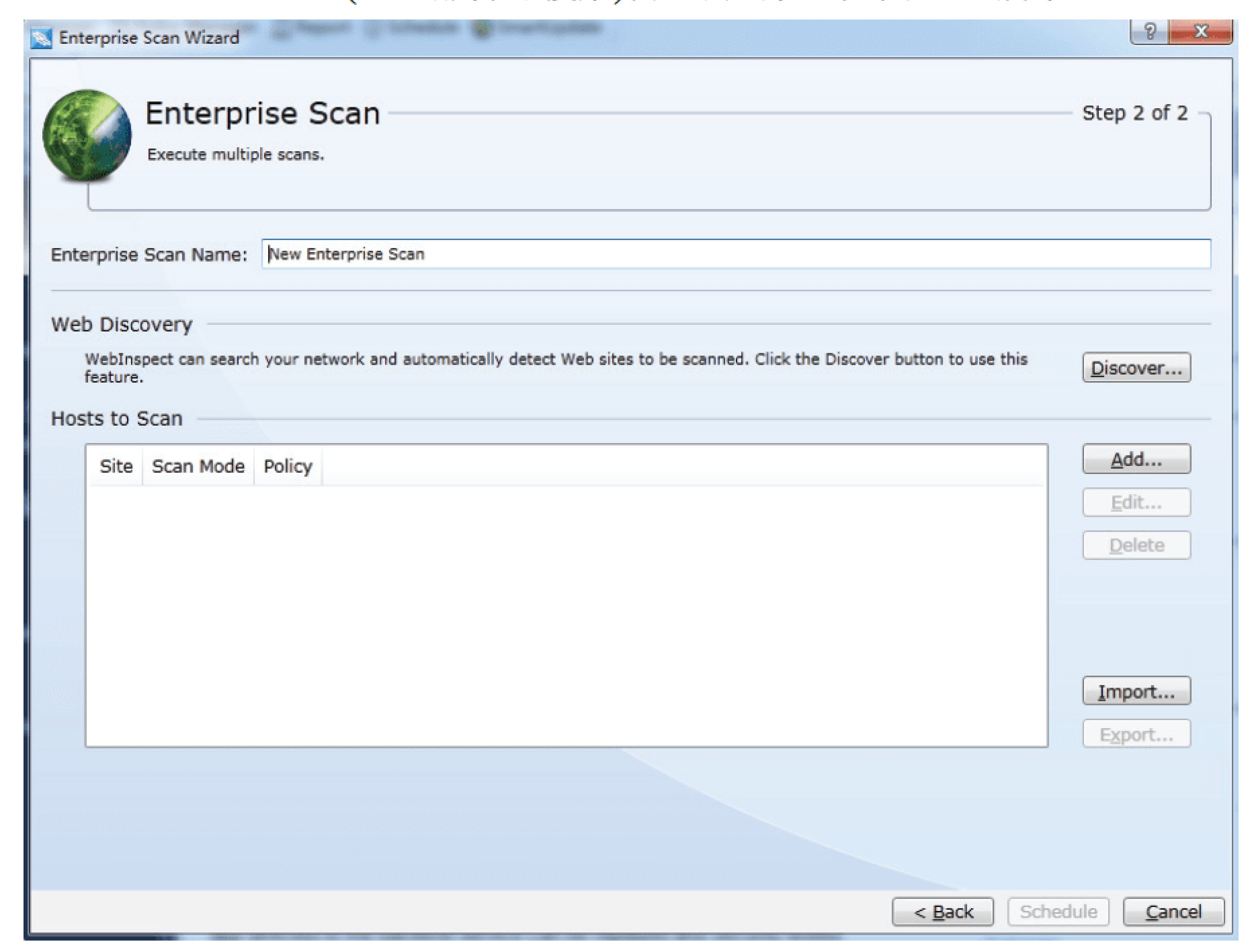



图 7-64　输入企业扫描信息

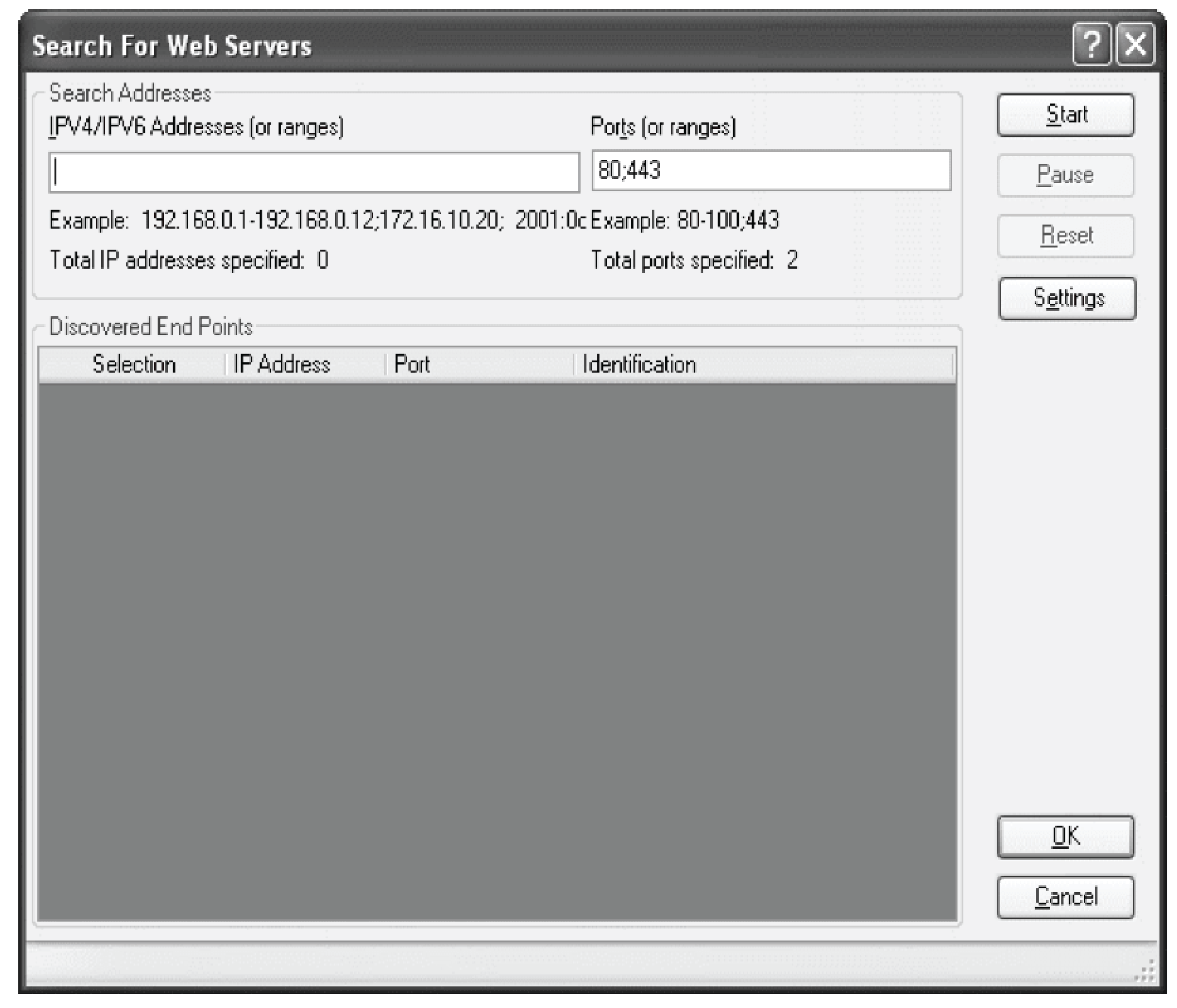



图 7-65　Search For Web Servers 窗口

2) 在 IPV4/IPV6 Addresses (or ranges)(IPV4/IPV6 地址(或范围))框中，键入一个或多个 IP 地址或 IP 地址范围。

- 使用分号分隔多个地址。

例如：172.16.10.3；172.16.10.44；188.23.102.5。

- 使用破折号或连字符分隔起始和结束 IP 地址范围。

例如：10.2.1.70-10.2.1.90。

注意，IPV6 地址必须括在括号中。

3) 在 Ports (or ranges)(端口(或范围))框中，键入要扫描的端口。

- 使用分号来分隔多个端口。

例如：80；8080；443。

- 使用破折号或连字符在一定范围内分隔起始和结束端口。

例如：80-8080。

4) 单击 Settings(设置)来修改套接字的数目和超时参数(可选)，用于发现可用服务器的过程。

5) 单击 Start(开始)，如图 7-66 所示。结果在发现终点(Discovered End Points)区域中显示。

- 单击 IP Address(IP 地址)列中的条目，以查看该网站。
- 单击 Identification(身份验证)列中的条目，打开会话属性窗口，在这里用户可以查看原始请求和响应。

6) 要从列表中删除服务器，请清除相应复选框中的 Selection(选择)列。

7) 单击 OK(确定)。

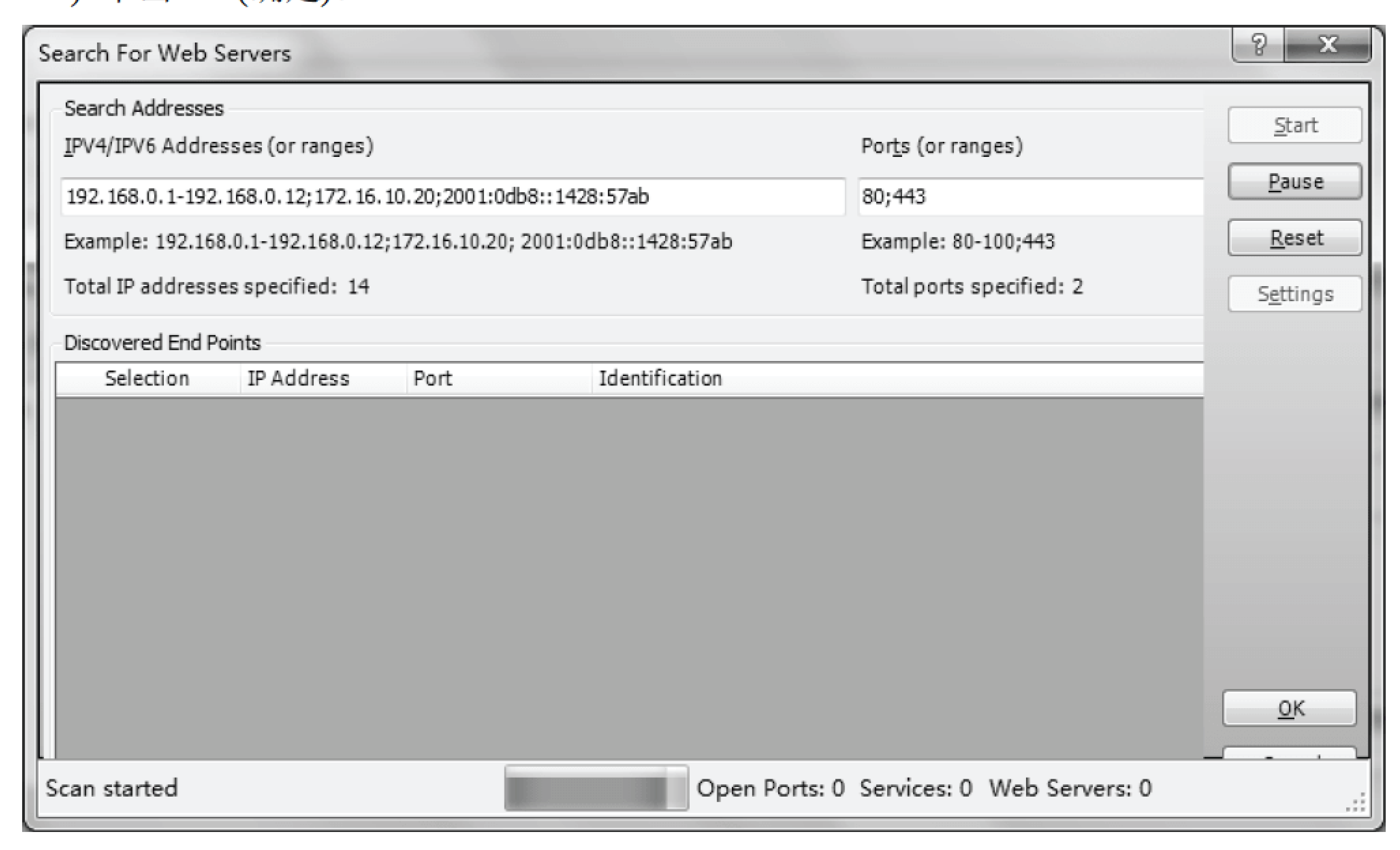



图 7-66　开始发现过程

主机扫描

按照下面的步骤手动输入要扫描的 URL 或 IP 地址列表。如图 7-67 所示。

1) 单击 Add(添加)。

2) 对 Web 网络添加描述信息。

3) 对其他服务器进行重复操作。

导入列表

如果以前使用企业扫描功能或网络发现工具来检测服务器，并以一个文本文件导出，用户可通过单击 Import(导入)加载这些结果，然后选择保存文件。

编辑“要扫描的主机”列表

建立使用一种或更多上面方法的服务器列表后，可使用以下步骤修改列表：

要修改特定的扫描设置：

1) 选择服务器。

2) 单击 Edit(编辑)。

3) 更改设置。

4) 单击 Finish(完成)。

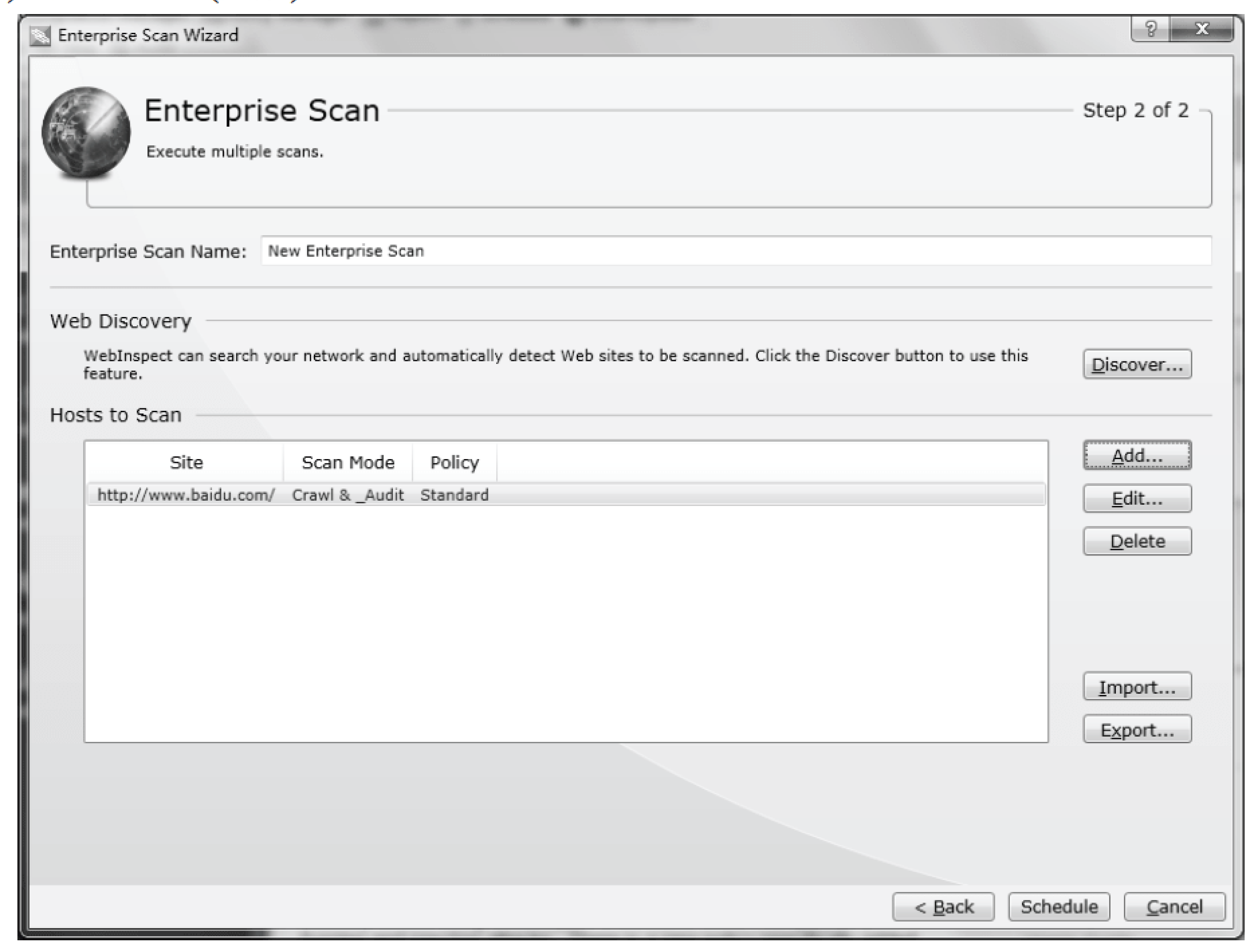



图 7-67　手动输入要扫描的 URL

要从列表中删除服务器：

1) 选择一个服务器。

2) 单击 Detele(删除)。

导出列表

要保存“要扫描的主机”的列表：

1) 单击 Export(导出)。

2) 使用一个标准的文件选择窗口，指定文件名和位置。

开始扫描

要开始企业扫描，单击 Schedule(计划)。每个服务器的扫描结果完成后，自动保存在用户默认扫描的文件夹中。服务器的名称以及日期和时间戳都被包含在文件名中。WebInspect

会提示已经安排在某个时间进行扫描，如图 7-68 所示。

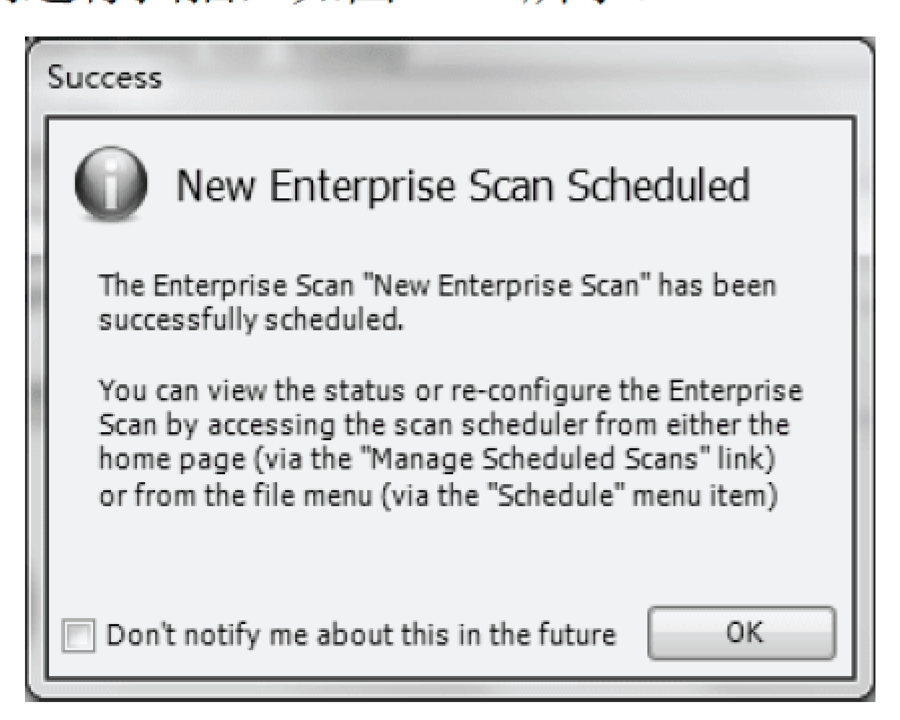



图 7-68　企业扫描设置完成

注意，WebInspect 许可证允许用户扫描特定的 IP 地址或一个地址范围。如果服务器有一个 IP 地址不被许可证允许，该服务器将不包含在扫描中。

## 7.3.5　手动扫描

手动扫描(也称为步模式)是一个网站的扫描选项，可让用户手动点击应用程序可访问的任何部分。它不会爬行整个网站，但会记录用户遇到的手动点击网站的这些资源的信息。此功能最常用的是通过一个登录页面进入网站或定义要调查该应用程序的离散子集，如图 7-69 所示。

一旦完成浏览网站，用户可审计结果来评估访问和记录方面的安全漏洞。

按照下面的步骤来记录使用步模式网站：

1) 在 WebInspect 菜单栏上，单击 File(文件)，再依次单击 New(新建)和 Basic Scan (基础扫描)。

或者，也可以单击 WebInspect 开始页选项卡，然后选择 Start a Basic Scan (启动基础扫描)。

2) 按照基础扫描描述，选择 Manual(手动)作为扫描方法，配置一个网站扫描的说明。

3) 配置所有扫描参数后，单击 Scan(扫描)。

4) 当浏览器打开时，用它来浏览目标网站，访问想记录的区域。

5) 完成后，返回 WebInspect 窗口，然后单击 Finish(完成)。

6) 当导航窗格返回到网站视图时，可单击 Audit (在工具栏上)来审计网站。

如果使用步模式来浏览一个网站，需要基本身份验证或 NTLM 认证，WebInspect 将记住用户名和密码，并当审计该网站时使用它们。每个服务器都可以有自己的认证方案；WebInspect 将步模式期间遇到的每个服务器上的 NTLM 和基本身份验证进行区别。

## 7.3.6　审阅和重新测试漏洞

WebInspect 提供几种方法来检查或重新测试发现的漏洞。

- 重新测试个别的漏洞
- 重新测试所有发现的漏洞
- 重新扫描整个网站
- 比较两次扫描




图 7-69　使用步模式网站

### 1. 重新测试个别的漏洞

重新测试功能是一个非常强大的工具，它确认开发商已经固定一个特定漏洞，而不必进行一个全新的扫描。

1) 打开扫描。

2) 在导航窗格右键单击一个漏洞会话，或在摘要窗格中的 Vulnerability(漏洞)选项卡上右键单击一个漏洞。

3) 从快捷菜单中选择 Review Vulnerability(审阅漏洞)。

4) 在漏洞审阅窗口中，单击 Retest(重新测试)。

WebInspect 可将所有漏洞路径重新提交到服务器，并对原始响应(重新提交前)的每个结果进行比较，并显示相匹配的原始重新测试响应的百分比。这表明路径漏洞是否被准确地再现。每个 HTTP 请求和响应的原始会话和重新测试会话可并列比较，立刻显露任何显著的变化。一旦该项目已被确认为一个漏洞，用户可将这个缺陷提交到 HP 质量中心(ALM)或 IBM Rational ClearQuest。

### 2. 重新测试所有发现的漏洞

在这种类型的扫描检查中，原始扫描中只会发现部分目标网站的漏洞。WebInspect 不进行新网站的爬行，而只是追溯漏洞会话(记录在原始扫描)和攻击资源，并使用先前所采用的相同方式。

使用在摘要窗格中的 Vulnerability(漏洞)选项卡以查看结果。该选项卡包含一个名为 Repro(重现)的额外列，其中可能包含以下值：

- 未发现/固定(Not Found/Fixed)：在原始扫描检测到的漏洞，通过重新测试未找到。这些漏洞会显示灰色文本。用户可以对这些项目的漏洞进行审阅和重新测试。括号中的百分比表示对这个判定的一个启发式的置信水平。
- 再现(Reproduced)：原始扫描和重新测试检测到相同的漏洞，即该漏洞仍然存在。

- 新建(New)：重新测试检测到，但不是原始扫描报告中的漏洞。

重新测试所有漏洞：

1) 执行下列操作之一：

- 打开扫描。
- 选择管理扫描页面上的扫描。

2) 单击 Rescan(重新扫描)，并选择 Verify Vulnerabilities(验证漏洞)。

### 3. 重新扫描网站

重新扫描功能可以轻松地从一个打开的(或选定的)扫描转换到原始扫描预装设置的扫描向导。用户可能希望更新网站(和原始扫描使用相同的设置)，以确定先前是否发现的漏洞已经被修复和一个新的漏洞是否被发现。或者，可能想调整一些设置来改善爬行或审计。

1) 执行下列操作之一：

- 打开扫描，单击工具栏的 Rescan(重新扫描)按钮，然后选择 Scan Again(再次扫描)。
- 在 WebInspect 开始页面，单击 Manage Scans(管理扫描)；然后选择一个扫描和单击 Rescan(重新扫描)。

2) 使用扫描向导，可选择修改用于原始扫描的设置。

注意：扫描名称的默认设置为<original_scan_name>-1。如果进行重新扫描，追加到默认名称，整数部分的值递增。

3) 在扫描向导的最后一步，单击 Scan(扫描)。

### 4. 比较扫描

可通过比较同一目标的两个不同扫描来发现漏洞，并使用这些信息，如图 7-70 所示。

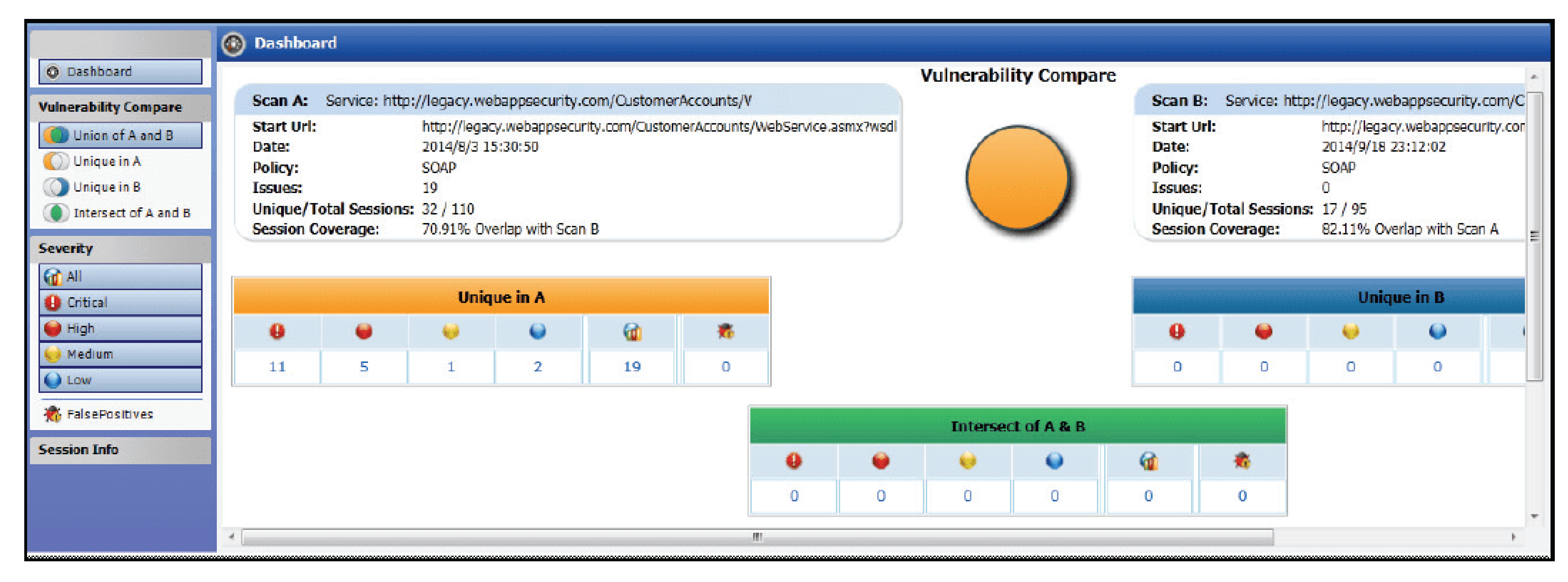



图 7-70　比较扫描结果图

- 验证补丁：漏洞被确定时，比较初始扫描检测到的漏洞和后续网站扫描到的漏洞。
- 检查扫描健康：更改扫描设置，这些变化扩大了攻击面的验证。
- 发现新漏洞：确定是否有新的漏洞已经在网站的更新版本中推出。
- 调查问题：追求异常现象，如误报或遗漏的漏洞。
- 比较授权访问：使用两个不同的用户账户来发现漏洞的执行扫描，该漏洞对于两个账户是独有的或共有的。

注意：两个扫描数据必须存储在同一个数据库类型(SQL Server Express 与 SQL Server Standard/Enterprise)。

请执行下列操作之一来比较两次扫描：

- 从管理扫描页面，选择两个扫描和单击比较。
- 从含有打开扫描选项卡中，单击 Compare(比较)；然后从扫描比较对话框的列表中选择扫描和单击比较。

如果选择扫描有不同的起始网址，或者如果使用不同的扫描策略，或者如果扫描是不同的类型(如一个基础扫描与一个 Web 服务扫描)，应用程序将出现一条警告消息，可以选择继续，也可以终止功能。

仪表板

对于扫描 A 和扫描 B，Vulnerability Compare(漏洞比较)选项卡的仪表板显示以下信息：

- 扫描 A / B：扫描的名称。
- 起始 URL：目标网站的网址。
- 日期：执行原始扫描的日期和时间。
- 策略：用于扫描策略。
- 问题：检测到的漏洞和误报的数量。
- 独有/会话总数：此扫描创建了独有的会话数量，相比于会话总数。
- 会话覆盖：常见两种扫描的会话百分比。

仪表板描绘两个集合的维恩图(也叫文氏图)：通过扫描 A(黄色圆圈表示)的漏洞发现和扫描 B(蓝色圆圈表示)的漏洞发现。两个集合的交集是由绿色的重叠表示。该图进行缩放，以反映两个集合之间的实际关系，如图 7-71 所示。

仪表板还显示独有的漏洞数量，按严重程度，由每个单独扫描和那些由两个扫描(交集)检测发现，用柱状图表示这些结果，如图 7-72 所示。

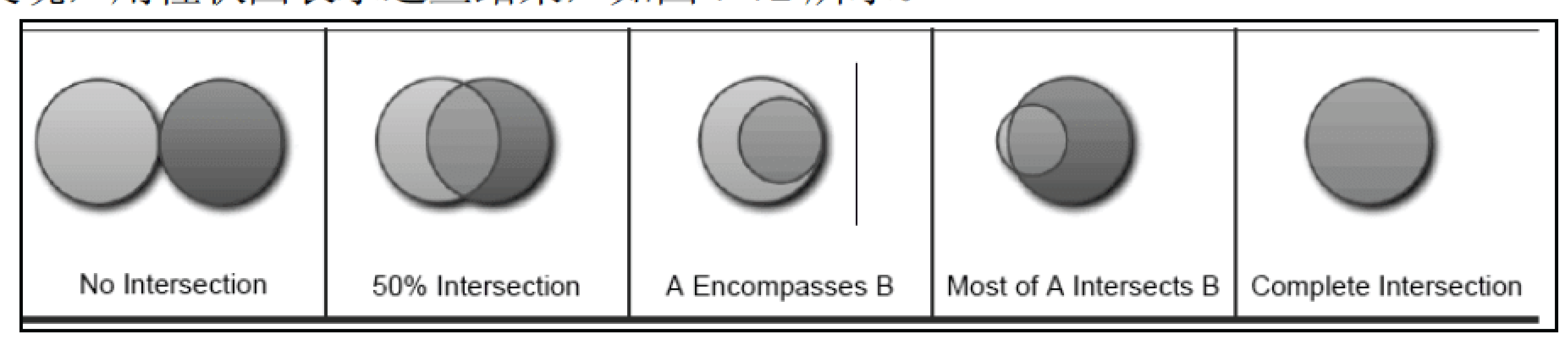



图 7-71　两个集合的维恩图

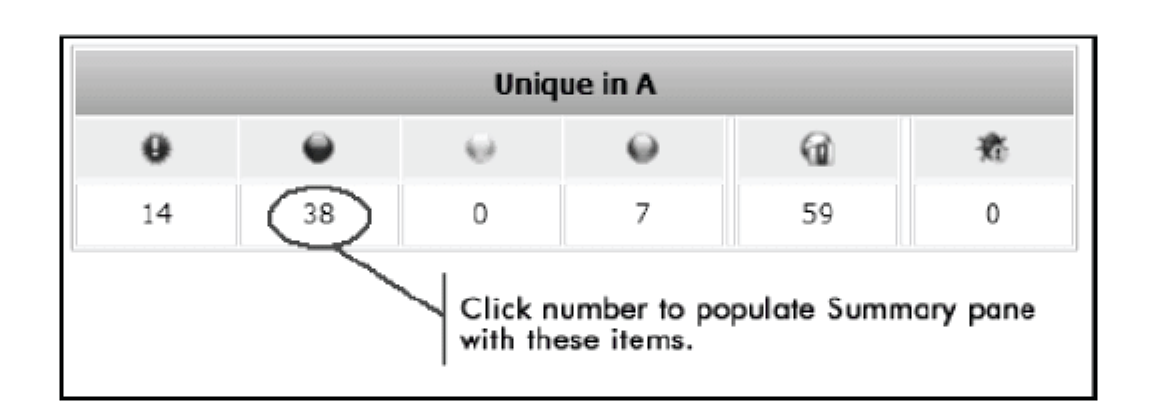



图 7-72　A 中唯一漏洞的数量

漏洞比较

在摘要窗格中显示一个或两个扫描检测到的漏洞列表，从漏洞中选择一个过滤器比较面板。选项包括：

A 和 B 的并集。

A 独有的。

B 独有的。

A 和 B 的交集。

也可以单击维恩图的相应部分显示 A 独有的和/或 B 独有的所发现漏洞。

注意，当比较扫描时，WebInspect 忽略主机和端口。考虑到运行在不同服务器的两个重复的网站。一个正在开发的网站可能在http://dev.mysite.com运行，而同一个网站可能会在 http://QA.mysite.com 来进行测试。当比较两个扫描之间的会话和漏洞时，不考虑主机的 URL 和端口。

例如：

扫描 A：会话 1 是一个 http 到 url http://qa.mysite.com/stuff/page/info.asp的请求。它有一个漏洞。

扫描 B：会话 1 是一个 http 到 url http://dev.mysite.com:8080/stuff/page/info.asp的请求。它和扫描 A 中的会话 1 具有相同的漏洞。

当比较扫描时，会话 1 会显示为“A 和 B 相交”。

严重性

从严重性面板根据严重性选择一个或多个选项来进一步过滤漏洞列表。选择是取并集。

注意，也可以分组和过滤结果。

默认情况下，网格显示以下栏目：路径、方法、漏洞号参数和扫描名称。要添加或删除列，请右键单击列标题栏并选择 Columns(列)。可用列如下：

- 严重性(Severity)：该漏洞的相对评价，从低到关键。请参阅下面的相关图标，如图 7-73 所示。
- 检查(Check)：WebInspect 探测特定的漏洞，如跨网站脚本、未加密的登录表单等。
- 路径(Path)：指向资源的完整路径。
- 方法(Method)：HTTP 方法，如 GET、POST 等。
- 堆栈(Stack)：从 SecurityScope 获得堆栈跟踪信息。此列当且仅当扫描 SecurityScope 被安装在目标服务器上时可用。
- 漏洞号参数(Vuln Param)：有漏洞的参数名称。
- 参数(Parameters)：分配的参数和值名称。
- 手动(Manual)：如果该漏洞被手动创建，显示一个复选标记。
- 重复(Duplicates)：通过 SecurityScope 检测到的漏洞可以追溯到同一个源头。
- 位置(Location)：路径加上参数。
- CWE ID(CWE ID)：与漏洞相关的共同弱点枚举标识符。
- 扫描名称(Scan Name)：扫描 A 和/或扫描 B。

漏洞的严重性由图 7-33 中所示的图标表示。

| Critical | High | Medium | Low |
| --- | --- | --- | --- |
| | | | |

图 7-73　漏洞严重性图标

随着会话被选中，也可以通过从会话信息面板中选择一个选项，查看相关信息

对于发送和查询参数，单击参数列中的条目显示参数的一个可读性概要。

FilesToURLs 工具

作为正常安装过程的一部分，WebInspect 安装两个命令行实用程序(FilesToURLs.exe 和 FilesToURLs.py)，旨在发现和评估所有网站资源。当在服务器上执行时，该实用程序会检查目标网站上的所有文件，并创建包含每个检测的 URL 的 XML 文件。然后，配置基础扫描时，可选择列表驱动的扫描方法，并提交该 XML 文件。表 7-11 列出 FilesToURLs.exe 的用法。表 7-12 列出 FilesToURLs.py 的用法。

注意，FilesToURLs.exe 需要 NET Framework 4.0 或更高版本。FilesToURLs.py 需要 Python2.6。

要创建 XML 文件并将其包含在一个扫描中：

1) 找到 FilesToURLs.exe(在 UNIX 系统中，找到 FilesToURLs.py)。默认位置是 C:\Program Files\HP\HP WebInspect。

2) 使用网络共享(或将文件复制到用户的 Web 服务器后)，运行该实用工具，根据下面的描述使用。

3) 启动 WebInspect。

4) 在扫描向导的第一步，选择 List-Driven Scan(列表驱动的扫描)。

5) 单击浏览按钮，选择由 FilesToURLs 实用程序生成的 XML 文件。

6) 完成向导并开始扫描。

表 7-11　使用 FilesToURLs.exe

| 参　　数 | 说　　明 |
|---|---|
| /docroot | Web 文件存储在本地路径(必需) |
| /outfile | 要创建的 XML 文件的名称(必需) |
| /include | 现有的文件，其内容应包含在输出中 |
| /hostname | 文件服务(默认值：本地主机名)的主机名 |
| /baseurl | 文件服务的基础 URL(默认是：/) |
| /port | 该 Web 服务器正在侦听的端口(默认：80 或 443) |
| /secure | 指示该端口正在使用 SSL |

表 7-12　使用 FilesToURLs.py

| 选　　项 | 说　　明 |
|---|---|
| -h, --help | 显示帮助消息并退出 |
| -d DOCROOT, --docroot=DOCROOT | Apache 的文档根目录或文件服务的其他目录 |
| -o FILE, --outfile=FILE | 输出到文件(默认为 STDOUT) |
| -i FILE, --include=FILE | 在输出中包含文件的内容 |
| -n HOSTNAME, --hostname=HOSTNAME | Web 服务器的主机名(默认为本地主机名) |
| -b BASEURL, --baseurl=BASEURL | 来自服务文件(默认为/)的基本 URL |
| -p PORT, --port=PORT | 服务正在侦听的端口(默认为 80 或 443) |
| -s, --secure | 使用 SSL 指定监听端口(默认为 false) |

该列表驱动扫描选项也可以使用手动创建的纯文本文件，而不是由 FilesToURLs 实用程序生成的 XML 文件。每行列出一个 URL。每个 URL 必须完全合法，并且必须包括协议(例如，http://或 https://)。

- IPV6
- WebInspect(从 8.1 版开始)支持 Internet 协议版本 6(IPv6)地址的网站和网络服务扫描。当指定起始 URL，则必须将 IPv6 地址写在括号内。例如：
  - 错误！超链接引用无效。
  - WebInspect 扫描“localho st”。
  - WebInspect 扫描主机开始于“子文件夹”目录的指定地址。

## 7.4　WebInspect 工具简介

一套强大的诊断和渗透测试工具集成在 WebInspect 中。包括：

- 审计输入编辑器
- Cookie Cruncher
- 编码器/解码器
- HTTP 编辑器
- 日志查看器
- 策略管理器
- 正则表达式编辑器
- 报表设计器
- Server 分析
- Server 事件探查器
- 智能更新
- SQL 注入
- SWF 扫描
- Web Brute
- 网络发现
- Web 窗体编辑器
- 网络 Fuzzer
- 网络宏录制
- Web 代理
- Web 服务测试设计

WebInspect 安装也包括 HP 支持工具，它提供了一种简捷的方法来上传文件，可帮助 HP 技术支持人员分析和解决用户在使用应用程序安全中心产品时遇到的任何问题。

当使用集成代理工具时，即使证书是必需的，用户也可能会遇到服务器不需要客户端证书的情况。在需要证书时，必须编辑 SPI.Net.Proxy.Config 文件。

步骤如下：

1) 打开 Microsoft Internet Explorer 中。

2) 单击 Tools(工具)，再单击 Internet Options(Internet 选项)。

3) 在 Internet 选项窗口中，选择 Content(内容)选项卡，然后单击 Certificates(证书)。

4) 在证书窗口中，选择一个证书，然后单击 View(查看)。

5) 在证书窗口中，单击 Details(详细信息)选项卡。

6) 单击 Serial Number(序列号字段)和复制出现在下部窗格中的序列号(突出的号码，然后按 Ctrl + C 键)。

7) 关闭所有窗口。

8) 打开要编辑的 SPI.Net.Proxy.Config 文件。默认位置是 C:\Program Files\HP\HP WebInspect(或用户自定义安装位置)。

9) 在客户端证书覆盖部分，添加以下条目：

<ClientCertificateOverride HostRegex="RegularExpression" CertificateSerialNumber="Number"/>

其中，RegularExpression 是一个匹配主机 URL 的正则表达式(例如：.*austin\.hp\.com).

10) 保存编辑的文件。

### 7.4.1　策略管理器

当使用 WebInspect 审计或爬行 Web 应用程序时，策略是审计引擎和攻击代理的集合。每个部件都有一个特定的任务，比如测试易感性跨网站脚本，网站建设树，探测已知的服务器漏洞等，这些组件被分为以下几组：

1) 审计引擎

2) 一般应用测试

3) 通用文本搜索

4) 第三方 Web 应用程序

5) Web 框架/语言

6) Web 服务器

7) 网站发现

8) 自定义检查

所有这些组件(除了审计引擎)统称为攻击群。每个攻击组所包含的单个模块(称为攻击代理)的子组检查用户网站的漏洞。

WebInspect 包含的几个预包装策略旨在满足大多数用户的需要。所有的策略包含所有可能的审计引擎和代理，但每个策略都有这些组件可使用的不同子集。用户可通过启用或禁用审计引擎和/或个人攻击代理(或代理组)来编辑策略。用户可编辑现有策略，也可用新名称保存并创建一个策略。

### 7.4.2　审计输入编辑器

有两种方法来访问审计输入编辑器：

- 从策略管理器(使用策略管理器 Tools(工具)菜单)。使用此方法来创建或修改一个输入文件(<filename>.inputs)。然后，用户可以修改扫描设置来指定该文件。要修改输入文

件，单击审计输入编辑工具栏的打开图标或选择 File(文件)，然后选择 Open(打开)。

- 从默认或当前设置(在攻击排除设置中单击审计输入编辑器按钮)。使用这种方法，用户可以直接修改默认设置，但不能创建一个单独的输入文件。

但是，如果从策略管理器访问审计输入编辑器，必须导入到 WebInspect 包含检查输入修改的保存文件。要做到这一点：

1) 在 WebInspect 菜单栏中，单击 Edit(编辑)，然后单击 Default Settings(默认设置)。
2) 选择 Attack Exclusions(攻击排除)。
3) 单击 Import Audit Inputs(导入审计输入)。
4) 选择用户所创建的文件，然后单击 Open(打开)。

当通过当前设置或默认设置窗口访问时，Attack Exclusions 面板、审计输入编辑器不包含菜单栏或工具栏。

### 7.4.3　Web 窗体编辑器

大多数 Web 应用程序都包含输入控件(文本框，按钮，下拉列表等)组成的形式。提交表单给代理处理之前，用户一般通过修改控制(如输入文本或检查框)“完成”表单。通常，这种处理会导致用户导航到另一页面或应用程序的一部分。例如，完成一个登录表单后，用户将进入到应用程序的开始页面。

有些网站(如 WebInspect 银行应用示例 zero.webappsecurity.com)包含许多不同的表单来完成各种交易。如果 WebInspect 是导航应用程序中所有可能存在的联系，它必须能够为每个表单提交适当的数据。

用户可以指定一个表单输入为“全局”，这意味着它的值将被提交给具有同名属性的任何输入控制，无论该 URL 发生在哪里。

在扫描过程中，如果 WebInspect 发现一个输入控件的名称属性和用户创建的文件不匹配，将提交一个默认值(12345)。

注意，如果用户使用的是代理服务器，Web 窗体编辑器将不会使用默认设置。必须首先配置 Internet Explorer 以便使用所需的代理。

有两种方法来创建表单值的列表：手动创建列表和通过应用程序浏览记录值。

### 7.4.4　Web Brute

此工具将确定用户名和密码是否很容易猜到。例如，如果一个客户访问网站时，使用 customer 用户名和 password 密码，你可能会想告诫该用户他的密码和用户名的脆弱性，并建议他改变密码和/或用户名。

Web Brute(网络狂人)将尝试登录表单或认证页面的“Brute Force”攻击，使用用户名和密码两个准备清单。

注意，这是一种侵入式攻击，并闯入到安全区域。Brute Force 攻击仅用于测试，而不应被用于对抗不知情的网站。

## 7.4.5　网络发现

使用网络发现在企业环境中找到所有打开的主机，如图 7-74 所示。

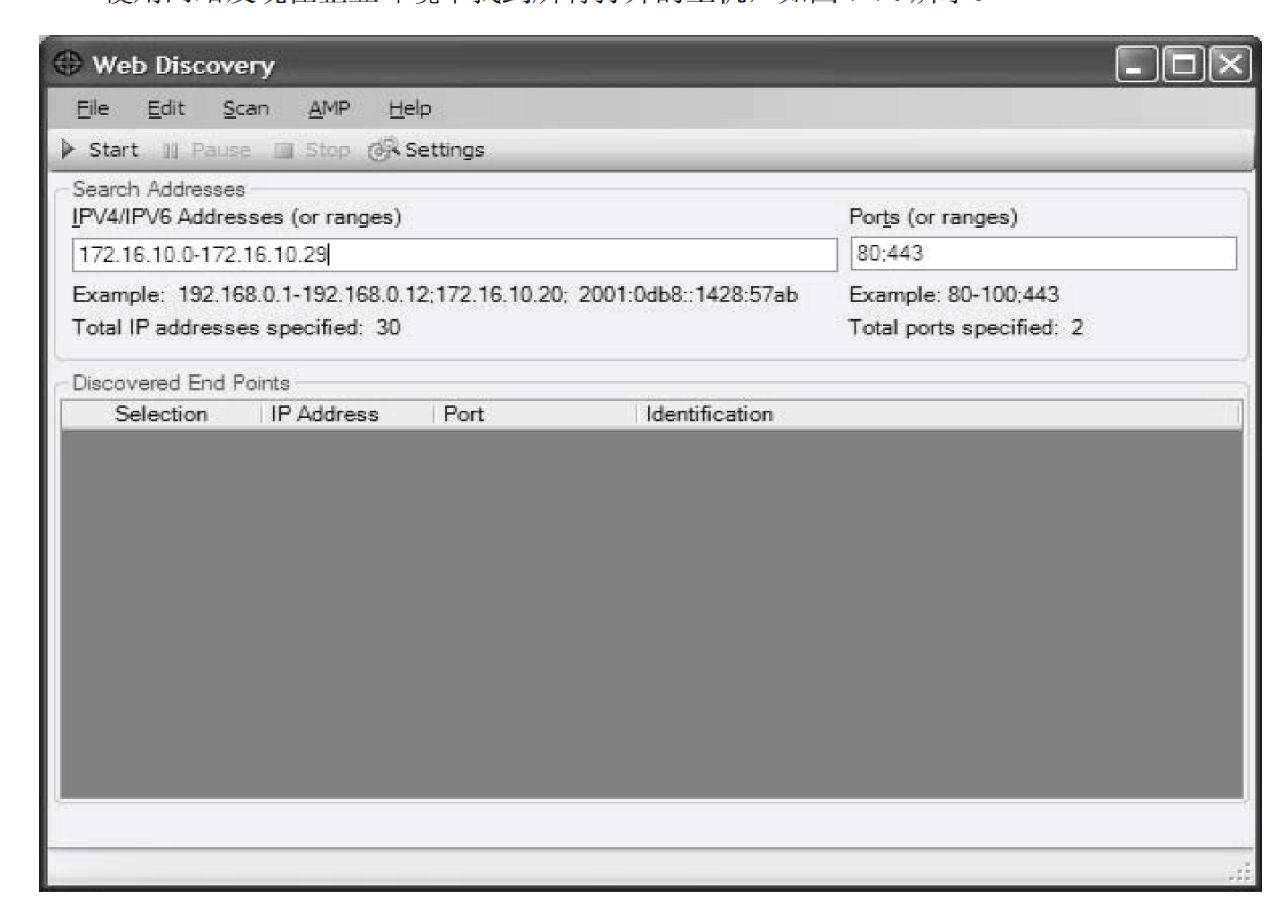



图 7-74　使用网络发现在企业环境中找到所有打开的主机

网络发现将数据包发送到所有开放的端口(一定范围内的 IP 地址和指定的端口)，搜索特定信息服务器的响应，然后显示结果。有包含网络发现的两个预定义数据包：Web 服务器和 SSL Web 服务器。它们都包含 GET / HTTP/1.0 的 HTTP 请求：网络发现用字符串“HTTP”搜索 HTTP 响应；如果找到该字符串，它显示 IP 地址、端口号以及文本“WebServer，”后跟一个旨在显示服务器的名称和版本号的正则表达式搜索结果。

用户可在文本文件中保存搜索的服务器列表。

## 7.4.6　编码器/解码器

该工具允许使用 Base64、十六进制、MD5 和其他方案来编码和解码值。也可以将一个字符串转换为 Unicode 字符串，并在 URL 中使用特殊字符。在扫描结果进行分析期间，当遇到怀疑之处时，编码或加密格式的字符串可以进行复制，将其粘贴到编码器/解码器工具，然后单击解码，如图 7-75 所示。

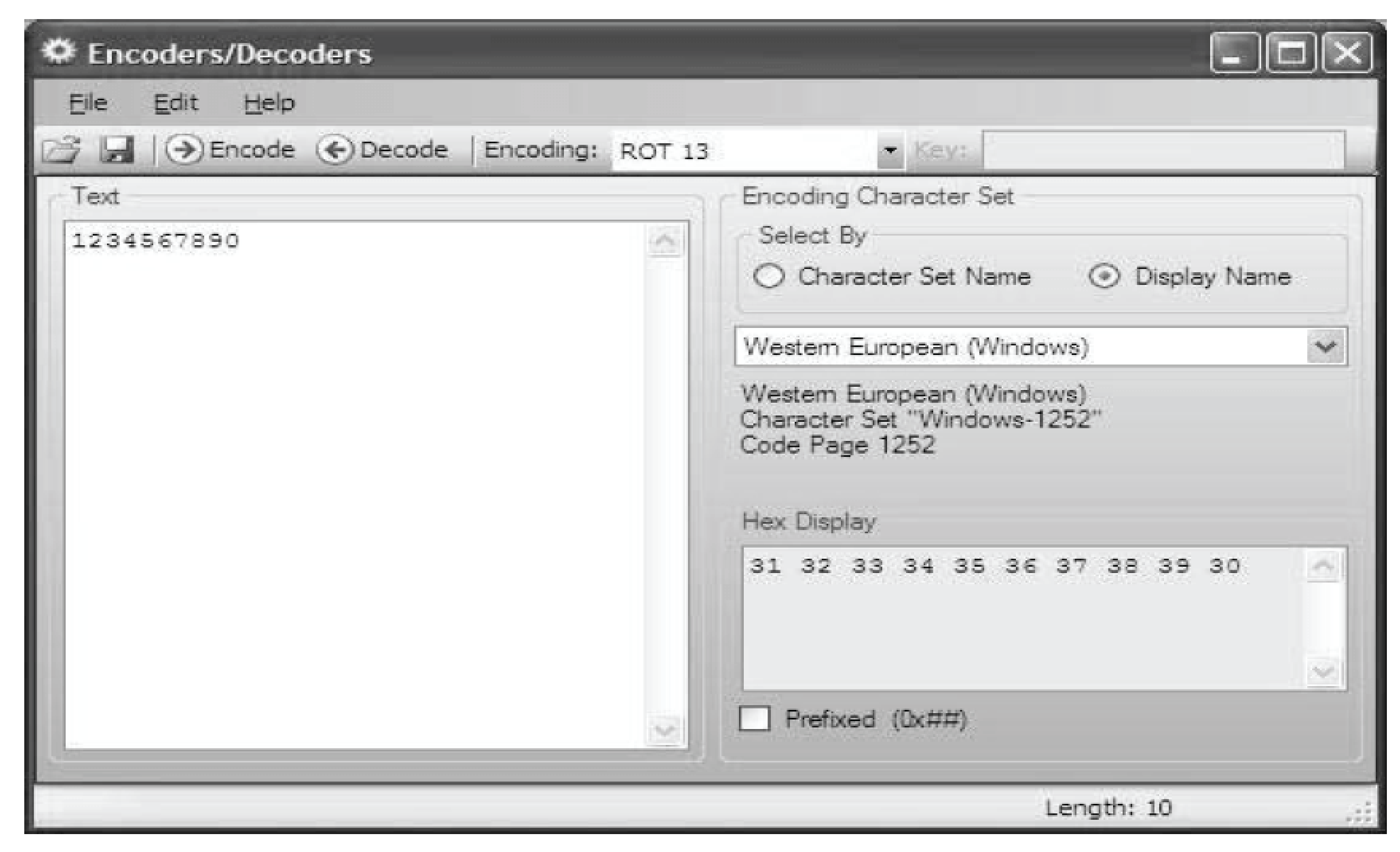



图 7-75　编码器/解码器工具

## 7.4.7　正则表达式编辑器

正则表达式是描述一组字符串的模式。正则表达式通过使用各种运算符来组合更小的表达式构造，类似于数学表达式。只有高级用户才能用正则表达式的知识使用此功能，如图 7-76 所示。

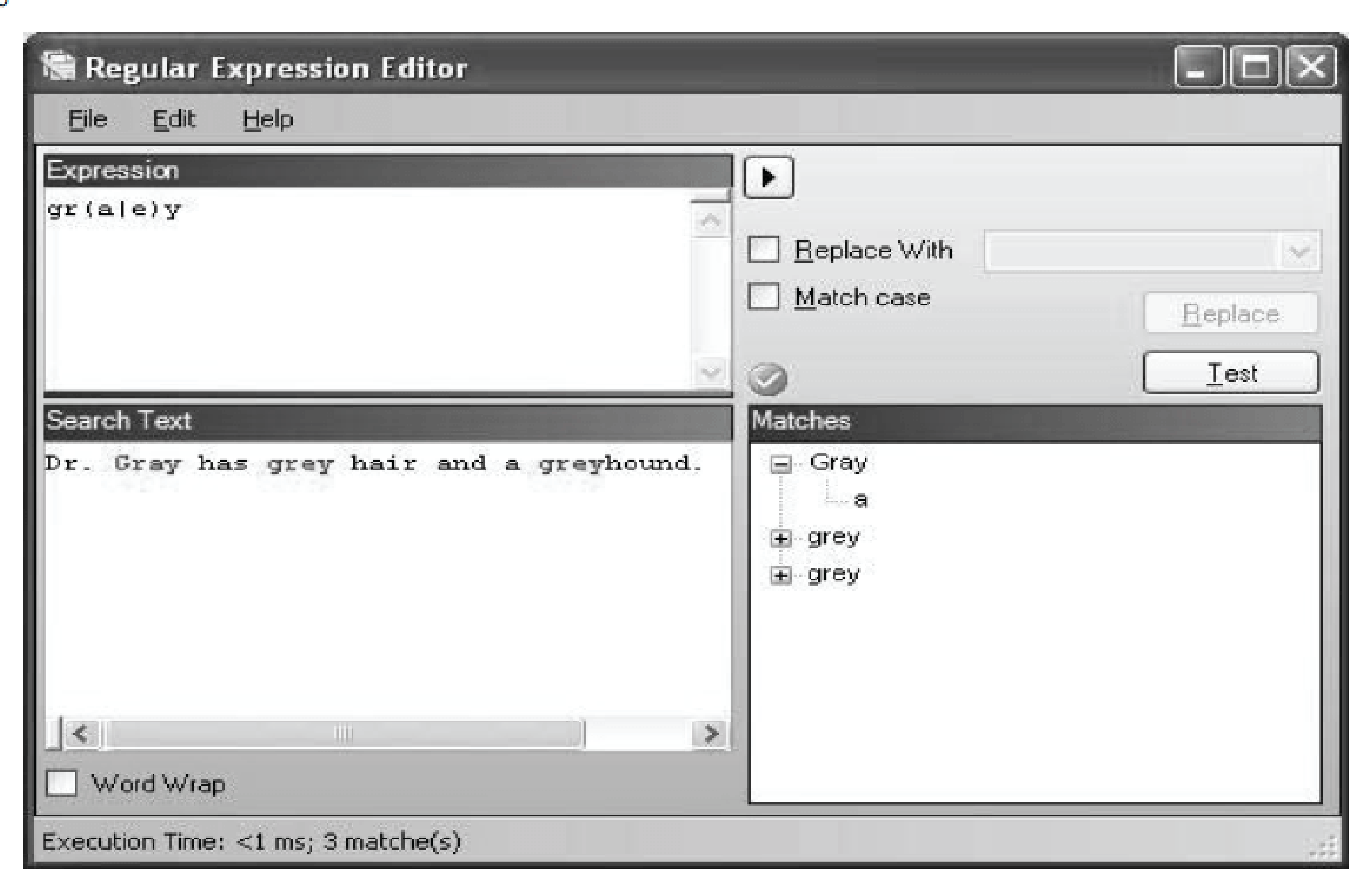



图 7-76　正则表达式编辑器

## 7.4.8　HTTP 编辑器

使用 HTTP 编辑器来创建或编辑请求，将其发送到服务器，并查看在原始的 HTML 或在浏览器中呈现的响应。

HTTP 编辑器是手动黑客工具，如图 7-77 所示。

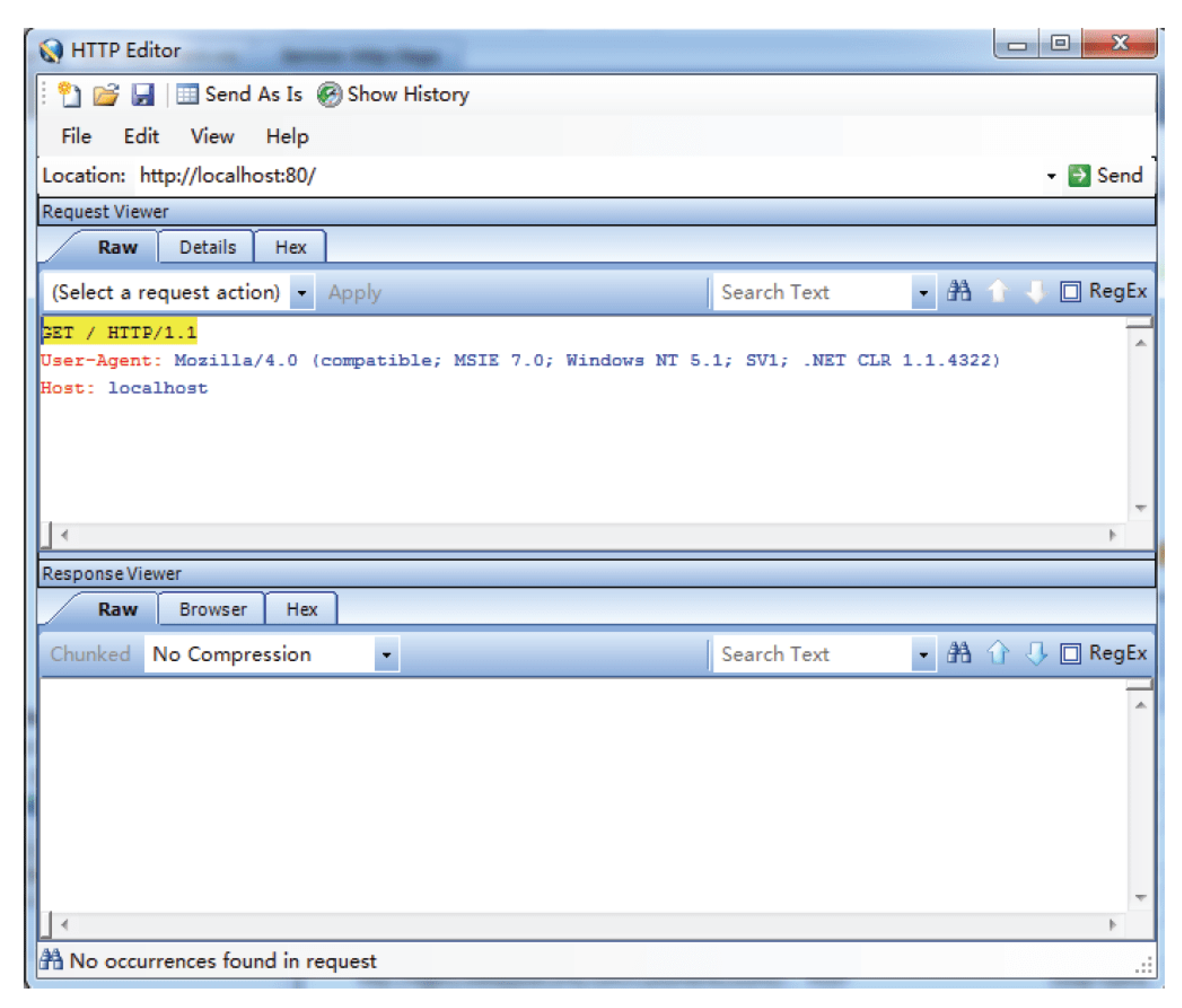



图 7-77　HTTP 编辑器

## 7.4.9　Web 代理

Web 代理是一个独立的代理服务器，可在桌面上配置和运行。

有了它，你可以从 WebInspect、浏览器、或其他工具监控流量。它是用于调试和渗透扫描的工具；当浏览一个网站，用户可以看到每个请求和服务器响应。

用户还可以创建启动宏或使用登录宏。

使用 Web 代理服务器与浏览器之前，必须配置浏览器的代理设置。如果使用 Internet Explorer：

1) 单击 Tools(工具)，再单击 Internet Options(Internet 选项)。

2) 单击 Connections(连接)选项卡。

3) 单击 LAN Settings(LAN 设置)。

4) 在 LAN(局域网)设置窗口中，选择 Use a Proxy Server for your LAN(为局域网使用代理服务器)，并输入代理服务器要使用的地址和端口。默认情况下，Web 代理服务器使用本地主机设置(127.0.0.1:8080)。

用户还应该在 Microsoft Internet Explorer 中通过代理连接使用 HTTP1.1。在 Internet Explorer：

1) 单击 Tools(工具)，再单击 Internet Options(Internet 选项)。

2) 单击 Advanced(高级)选项卡。

3) 在 HTTP 1.1 设置部分，选择 Use HTTP 1.1 through proxy connections(通过代理连接使用 HTTP 1.1)。

### 7.4.10　智能升级

使用智能更新下载最新的方案，以及漏洞和策略信息。智能升级也保证了使用 WebInspect 的最新版本，如果该产品的新版本可供下载也会提示。通过智能更新下载的新漏洞检查不会自动添加到创建的任何自定义策略。

注意，对于 AMP 安装，如果使用 WebInspect 智能更新更改或替换某个 AMP 的相关文件，检测部件服务可能会停止，检测部件会显示状态为“脱机”。必须启动 WebInspect 应用程序并重新启动服务。要做到这一点，从 AMP 的菜单中选择配置，然后单击 AMP 配置窗口的 Sensor Service(检测部件服务)选项卡上的 Start(开始)按钮。

1) 从工具栏中，单击Smart Update(智能更新)或从Tools(工具)菜单中选择Smart Update(智能更新)，或从 WebInspect 开始页选择启动 Smart Update(智能更新)。

2) 有可用更新时，智能更新窗口显示多达三个独立的可折叠面板来下载以下内容：

- 新的和更新检查
- WebInspect 软件
- 智能更新软件

选择一个或多个下载选项的关联复选框。

3) 确定要安装的更新，单击 Download(下载)。

如果还没有可用的 WebInspect 新版本，惠普将继续提供更新的知识库，但只有 10 天期限。超过这个时间，直到你下载新的 WebInspect 软件，更新才可用。

### 7.4.11　Cookie Cruncher

Cookie Cruncher 分析 Cookie 相对容易确定。

网页的超文本传输协议(HTTP)是无状态的，这意味着每个通信是离散的，不涉及发生的先后顺序。因为没有连续性固有的协议，应用程序设计人员推出了“会话”概念。当用户登录到应用程序中，会在服务器上创建会话，以维护来自同一个用户的该状态的其他请求。

每个会话都有一个唯一的标识符(会话 ID)。这个文本字符串在客户端和服务器之间传输，并且可以存储在 Cookie、URL 或网页的隐藏字段。与会话 ID 相关的一个问题是，许多网站使用的时间或 IP 地址算法具有可预测性。这种可预测性使得网站容易受到会话劫持。

### 7.4.12　网络 Fuzzer

Fuzzer(模糊测试)是生成和提交随机或连续的数据到应用的各个领域中，试图发现安全漏洞的自动化软件测试技术。例如，搜索缓冲区溢出的时候，测试者可以简单地生成各种大小的数据并将其发送到应用程序入口点之一，观察应用程序如何处理它。

网络 Fuzzer 可运行 Web 应用程序安全漏洞的常见几类自动化测试，包括 SQL 注入、格式字符串、跨网站脚本、路径遍历、缓冲区溢出以及协议的实施问题。

### 7.4.13　SQL 注入

SQL 注入是利用 SQL 查询中的客户端提供的数据，而不必首先删除可能有害字符的 Web 应用程序技术。SQL 注入支持 MS-SQL、Oracle、Postgress、MySQL 和 DB2 作为数据库类型，同时还支持多语言系统。

对于 SQL 注入漏洞工具测试，创建并提交 HTTP 请求可以由用户的 SQL 服务器处理。如果 Web 应用程序允许使用由用户提供的数据来更新或创建数据库记录，SQL 注入可能产生虚假的记录。为避免这种可能性，不测试生产数据库。相反，使用该数据库的副本，或使用不访问生产数据的测试账户，或排除任何可以更新或删除数据库中数据的页面。如果这些替代方案不可行，在当网站有很少(或根本没有)客户流量的一次测试之前，备份生产数据库。

### 7.4.14　合规经营

WebInspect 采用一个广泛的军火库(arsenal)的攻击代理，旨在检测基于 Web 应用程序中的安全漏洞。它侦测用户的系统与成千上万的 HTTP 请求，并评估个体反应。这种基于会话的扫描报告每个漏洞，精确定位其在应用中的位置，并建议应采取的纠正措施。这是对系统的定量分析。

它还能通过分级进行定性分析。例如，健康保险流通与责任法案(HIPAA)要求医疗服务提供者使用基于 Web 的应用程序提供“创建、更改和维护密码的步骤。”用户可评估自己的申请，然后生成合格报告，衡量应用程序如何满足 HIPPA 规则。

### 7.4.15　日志查看器

使用日志查看器检查由 WebInspect 维护的各种日志。此功能主要供 HP 产品支持小组调查报告事件。

### 7.4.16　流量模式的 Web 宏录制(过时)

宏是当访问并登录到一个网站上发生事件的记录。随后，用户可指示 HP 扫描仪使用该记录开始扫描。

WebInspect 10.0 版包括一个 Web 宏录制工具。它不包括在以前的版本中提供的流量模式的 Web 宏录制。但版本 10.0 的 Web 宏录制允许打开、播放和编辑以前的 WebInspect 版本中创建的现有流量模式宏，并创造新的流量模式宏，使用它的 Internet Explorer 浏览器技术(即 IE 技术)选项。当录制新宏时，在默认情况下，10.0 版本的 Web 宏录制首先使用基于事件的记录和 Firefox 浏览器技术，但如果由于某种原因失败，Web 宏录制会自动切换到访问流量模式的 IE 技术作为一种替代方法。基于它的多功能性，10.0 版本的 Web 宏记录也称为统一的 Web 宏录制。

### 7.4.17　基于事件的 IE 浏览器兼容的 Web 宏录制(隐藏)

WebInspect 10.0 版包括一个 Web 宏录制工具。默认情况下，它使用基于事件的功能和 Firefox 浏览器技术录制新宏。在以前的版本中提供单独的基于事件的 IE 浏览器兼容的 Web 宏录制无法再通过 WebInspect 菜单直接访问。实际上，它是隐藏的。然而，正如本节所述，10.0

版本的 Web 宏录制可以让用户间接打开、播放和编辑在以前的 WebInspect 版本中创建的基于事件的宏，也允许创建新宏。

注意：HP 强烈建议使用 WebInspect 10.0 版本的 Web 宏录制来记录所有新登录宏和工作流宏。

### 7.4.18　网络宏录制(统一)

用 Web 宏录制工具来录制登录宏。而 Web 宏录制是 WebInspect 10.0 版本提供的唯一直接进入宏录制的工具，它提供(或可访问)以前的 WebInspect 版本的三个 Web 宏录制的功能。基于它的多功能性，10.0 版本的 Web 宏记录也称为统一的 Web 宏录制。

Web宏录制可以采用多种启动方式，同时配置向导扫描或基础扫描。

被记录在一个基础扫描或向导扫描下的宏可以在任一类型的扫描中使用。

默认情况下，Web 宏录制操作使用相关的 Firefox 浏览器技术来记录和播放宏。它也可以使用 Internet Explorer 浏览器技术(IE 浏览器技术)来记录和显示网络流量数据。

### 7.4.19　Server 事件探查器

使用 Server 事件探查器来进行网站的初步审查，以确定某个 WebInspect 设置是否应该被修改。如果更改是必需的，Server 事件探查器返回一个建议列表，用户可以接受或拒绝。Server 事件探查器界面如图 7-78 所示。

例如，Server 事件探查器需要检测到授权才能进入网站，但用户没有指定一个有效的用户名和密码。与其执行将返回显著减弱结果的扫描，用户可按照 Server 事件探查器的提示，继续配置所需的信息。

同样，你的设置可能会指定 WebInspect 不应该执行“文件未找到”的检测。这个过程对于不返回一个“404 未找到”状态的网站十分有用，当客户端请求不存在的资源(可能代替返回状态“200 OK”，但响应中包含一条找不到该文件的消息)。如果 Server 事件探查器确定该计划已在目标网站实现，它会建议用户修改 WebInspect 设置，以适应此功能。

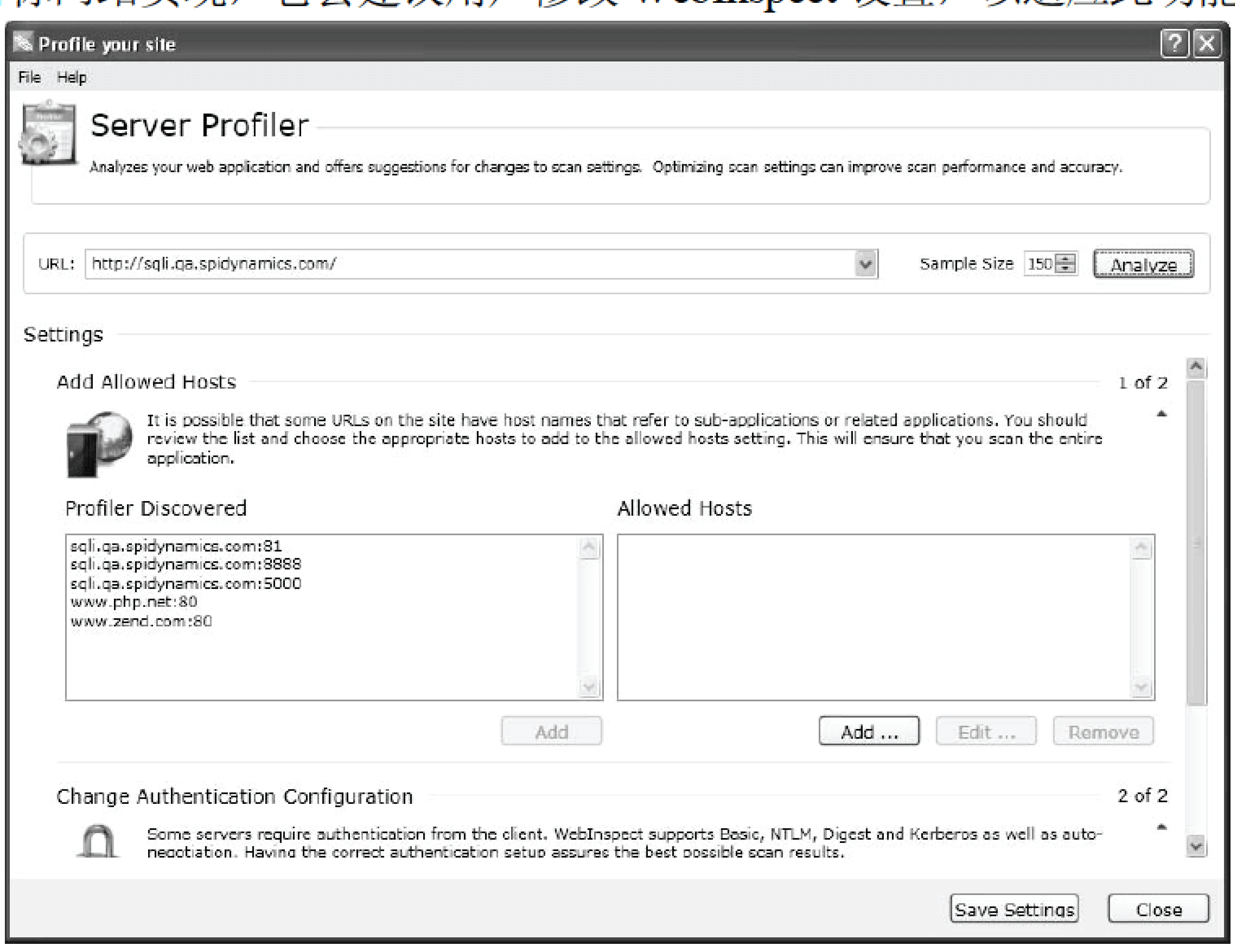



图 7-78　Server 事件探查器

### 7.4.20　Server 分析

Server 分析考查一个服务器，以确定服务器的操作系统、横幅、Cookie 和其他信息。Server 分析界面如图 7-79 所示。

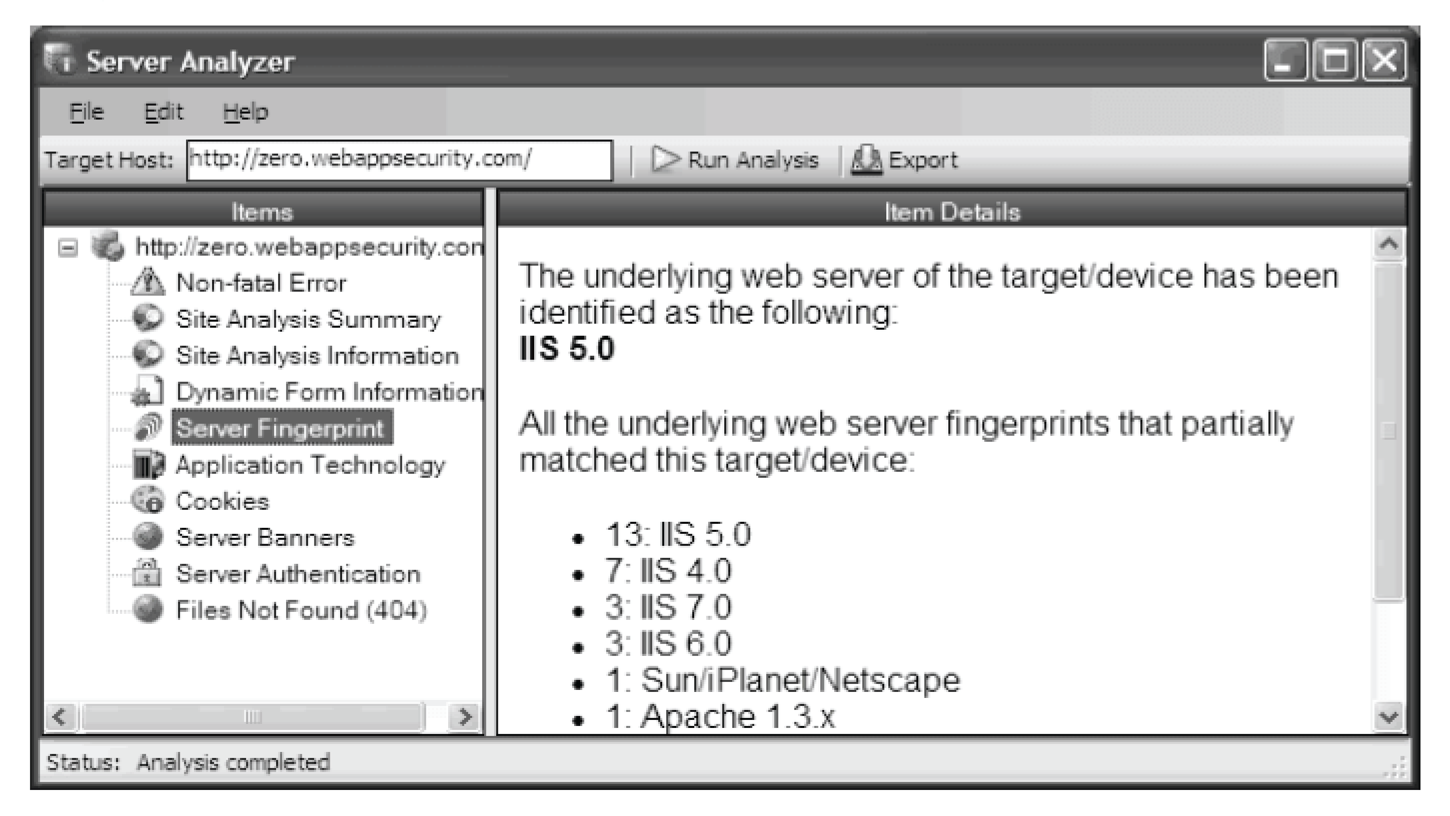



图 7-79　Server 分析

### 7.4.21　SWFScan

HP 网络安全研究小组开发 SWFScan，以帮助企业使用 Adobe Flash 平台开发安全的应用程序。这一创新工具识别许多影响 Flash 应用程序的漏洞，并明确说明如何消除或避免这些漏洞。

SWFScan 唯一支持所有版本的 Adobe Flash 和 ActionScript，其中包括 ActionScript2 和 3(Flash 版本 9 和 10)。

当 Internet 或 Intranet 上的 Flash 文件指向 SWFScan，或从本地计算机加载 Flash 文件时，SWFScan 反编译 SWF 字节码，生成 ActionScript 源代码，并进行静态分析。然后，可以生成报告。

WFScan 还提供了额外的关键信息(如网络电话、外部域的请求等)，对用户手动检查 Flash 应用程序十分有用。

### 7.4.22　报告设计器

报告设计器是由 Grape City - Data Dynamics 开发的 ActiveReports® 3.0 报告设计，并集成到惠普产品中。它具有创建和修改报告的功能。

报告设计器包含 6 个主要部件，如图 7-80 所示：

- 工具栏
- 设计器标签
- 工具箱
- 设计图面
- 报表资源管理器
- 属性网格

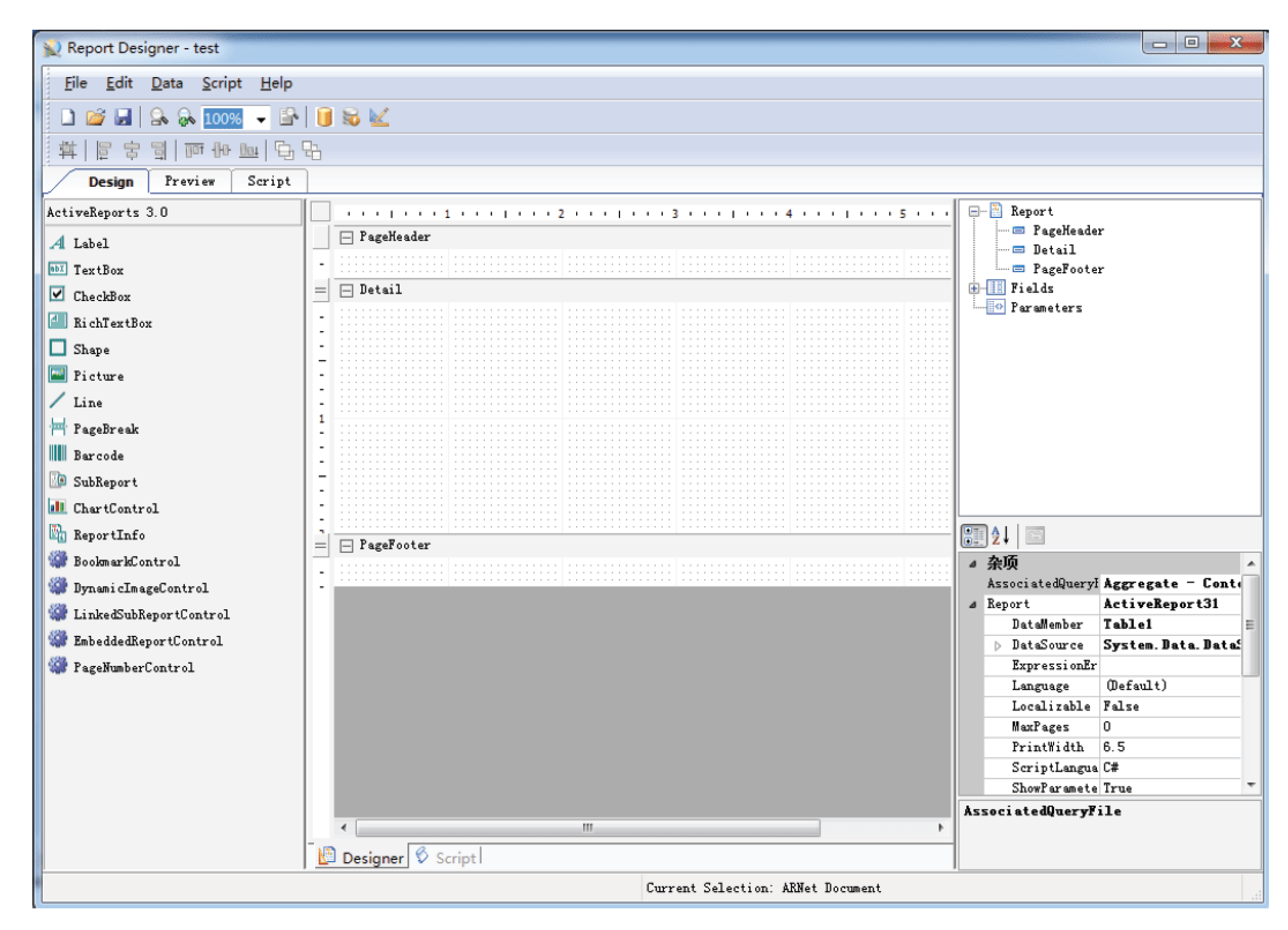



图 7-80　报告设计器

## 7.4.23　HP 支持工具

HP 支持工具提供了一种快速简便的方法来上传文件，可以帮助 HP 技术支持人员分析和解决在使用应用程序安全中心产品遇到任何问题。所有通信都使用 SSL 或 FTPS 协议。如图 7-81 所示。

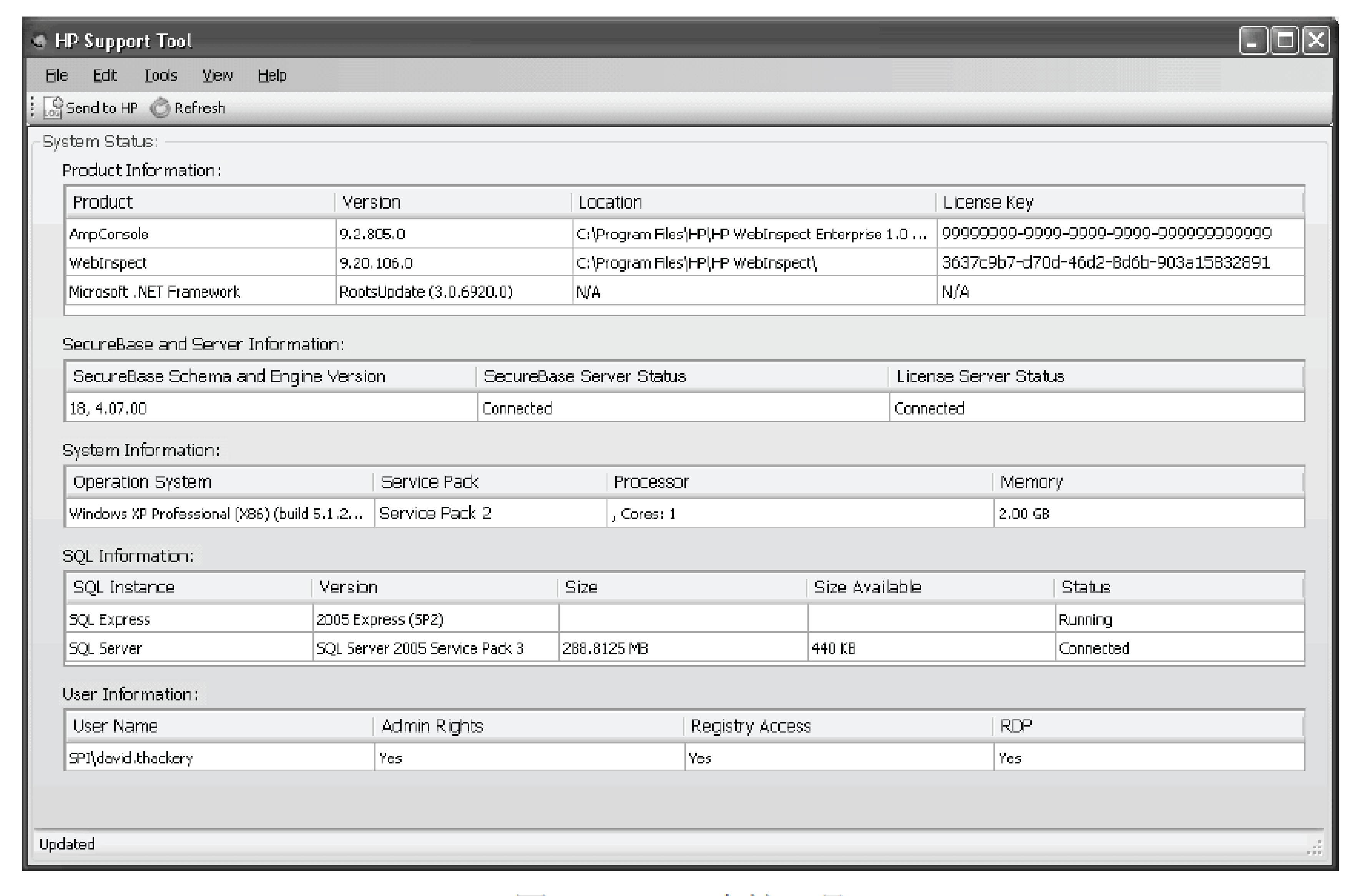



图 7-81　HP 支持工具

### 7.4.24 Web 服务测试设计

Web 服务是其他应用程序(而不是用户)和信息请求的应答通信程序。大多数 Web 服务使用简单对象访问协议(SOAP)发送 XML 数据。与 HTML 不同，它只是描述了网页的显示方式，XML 提供了一个框架描述，并包含结构化数据。客户端 Web 应用程序可以容易地理解所返回的数据，并将该信息提供给最终用户。

访问 Web 服务的客户端 Web 应用程序接收到一个 Web 服务定义语言(WSDL)文件，以便了解如何与该服务进行通信。WSDL 文档描述包含在 Web 服务编程的程序、这些程序期望的参数以及客户端 Web 应用程序将收到的返回信息类型。

使用 Web 服务测试设计来创建 Web 服务测试设计文件(filename.wsd)，该文件包含进行 Web 服务扫描时 WebInspect 提交的值。

## 7.5 本章小结

本章介绍了WebInspect的应用实践，主要分为三部分，第一部分介绍了 WebInspect 的基本按钮，包括按钮栏、菜单栏、工具栏以及其他一些组件。第二部分主要介绍了 WebInspect 的 5 大功能，是第一部分基本按钮的综合使用，是该工具的应用核心，包括向导扫描、基础扫描、Web 服务扫描、企业扫描以及生成报告，通过基础扫描、Web 服务扫描、企业扫描可以对 Web 应用程序和 Web 服务进行漏洞评估；生成的报告可以对漏洞的分布以及漏洞代码有更全面的直观认识，也可以直接将此报告提交给网站开发人员，让他们根据网站的实际情况对漏洞进行修复。第三部分是对 WebInspect 一套强大诊断和渗透测试工具的简述，这些工具集成在 WebInspect 中，使用它们能更方便地对结果进行分析。

# 第8章　HP Fortify工具使用

## 本章导读

Fortify 静态代码分析器(Static Code Analyzer)是HP Fortify公司开发的一个静态的、白盒的软件源代码安全测试工具。它通过内置的五大主要分析引擎对应用软件的源代码进行静态分析，期间与它特有的软件安全漏洞规则集进行全面匹配，从而将源代码中存在的安全漏洞扫描出来，并提供整理报告和修改建议。

另外，本章还将介绍实时安全分析器(RTA)和程序跟踪分析器(PTA)的基本知识。

应掌握的知识要点：

- HP Fortify 静态代码分析器的主要特征；
- 软件安装；
- 静态代码分析器分析原理；
- 静态代码分析器分析过程；
- 静态代码分析器扫描的方式；
- 静态代码分析器转换源代码；
- Audit Workbench 用户指南；
- 实时安全分析器基本知识；
- 程序跟踪分析器基本知识。

## 8.1　HP Fortify 静态代码分析器的主要特征

### 8.1.1　Fortify SCA 概述

Fortify SCA (Static Code Analyzer)是一组软件安全分析器，能够在多种语言中搜索那些违背安全编码规则和指导原则的情况。由Fortify SCA提供的丰富数据能够使这些分析器查明违背规则的情况并将其分级，从而可以快速准确地进行修复。这样不仅可以提供更安全的软件产品，还有助于使安全代码的检查更加有效、一致和完整，特别是涉及大型代码库的情况。模块化的体系结构能够快速地上传新的、第三方的以及客户指定的安全规则。

概括来讲，使用Fortify SCA 会涉及以下操作：

- (可选操作)将Fortify SCA 集成到构建过程中。
- 针对代码库执行分为两个阶段的分析过程，最后产生一份安全漏洞报告。
- (可选操作)将结果传送到Audit Workbench 和Fortify Team Server，进行分析与检查。

### 8.1.2　分析器概述

Fortify SCA 包含 6 个不同的分析器：数据流分析器、控制流分析器、语义分析器、结构分析器、配置分析器以及缓冲分析器。每个分析器均会接受不同类型的规则，该规则经过特殊定制，可为相应类型的分析提供必要的信息。规则是指用来识别源代码中可能引发安全漏洞或其他不安全因素的各种元素的定义。规则根据所适用的分析器进行划分，于是就产生了针对数据流分析器、控制流分析器、语义分析器、结构分析器、配置分析器以及缓冲分析器的各种规则。规则类型还可以进一步划分，以反映问题的类别或者规则所代表的信息类型。

表 8-1 列出并介绍了每种 Fortify 源代码分析器。

表 8-1　Fortify 源代码分析器

| 分　析　器 | 描　　述 |
|---|---|
| 数据流 | 数据流分析器可检测那些将被感染数据(用户控制的输入)用于危险用途的潜在漏洞。数据流分析器使用全局的、程序间的感染繁殖分析方法来检测source(用户输入的站点)与 sink (危险的函数调用或者操作)之间的数据流。例如，数据流分析器可以检测一个用户控制的特别长的字符串输入是否正在被复制到一个固定长度的缓冲区中，还可以检测一个用户控制的字符串是否正被用来构建 SQL 查询文本 |
| 控制流 | 控制流分析器可以检测潜在危险的操作执行顺序。通过分析程序中的控制流路径，控制流分析器能确定在执行一系列操作时是否遵循了特定的顺序。例如，控制流分析可以检测 check/time 的时间和未经初始化的变量，并检查实用程序(如 XML 阅读器)是否在使用前做了正确的配置 |
| 语义 | 语义分析器可在程序内部这个层面检测可能会引发潜在危险的函数和 API 的各种使用情况。其特定的逻辑会搜索 buffer overflow、formatstring 和各种执行路径的问题，但并不局限于这几个类别。任何存在潜在危险的函数调用都可以通过语义分析器进行标记。例如，语义分析器可以检测 Java 中的过时函数和 C/C++ 中的不安全函数，如 gets() |
| 结构 | 结构分析器可检测可能存在危险的结构缺陷或程序定义。通过理解程序构建的方式，结构分析器能够识别出安全编程实践中违反规则的各种情况以及通常难以被检测到的技术，原因是这些技术涉及的范围很广，其中包括有关声明和变量函数的使用。例如，结构分析器能检测在 Javaservlets 中成员变量的赋值，识别未被声明为 static final 的记录器的使用，并标记那些由于被断言为错误而永不被执行的 dead code 实例 |
| 配置 | 配置分析器可在应用程序的配置文件中搜索错误、缺陷和违反规则的策略漏洞。例如，配置分析器可检查网络应用程序的某一用户会话的超时是否合理 |
| 缓冲 | 缓冲分析器检测缓冲区溢出漏洞，其中漏洞涉及写/读的数据比缓冲区可以保存的数据更加丰富。缓冲区可以基于栈分配或堆分配。缓冲分析器使用程间分析来确定是否有一个条件导致缓冲区溢出。如果所有通往缓冲区的执行路径都导致缓冲区溢出，SCA 会报告这是缓冲区溢出漏洞并指出导致漏洞的变量。一旦有通往缓冲区的执行路径导致缓冲区溢出或变量值导致缓冲区溢出问题(用户辅助控制)， SCA 也都会跟踪数据流踪迹报告并显示变量是如何受破坏 |

# 8.2　软件安装

## 8.2.1　安装组件概述

Fortify SCA 安装包含下列一个或多个组件：

- Audit Workbench
- Fortify SCA
- Fortify Team Server
- Eclipse 3+ 安全编码插件
- 用于 Microsoft Visual Studio 2003 的安全编码包
- 用于 Microsoft Visual Studio 2005 的安全编码包

从 Fortify SCA 5.0 开始，Developer、Team 和 Enterprise 版本不再包含以上组件。

## 8.2.2　下载软件

Fortify SCA 软件和许可证文件可从 Fortify Customer Portal 下载。用户会收到包含客户账户信息(如用户名和密码)的确认邮件。

Fortify Customer Portal 的 URL 为 https://customerportal.fortify.com。

将软件包(包括许可证文件)下载到硬盘。还可从 Fortify Customer Portal 下载安全编码规则包，然后使用规则包更新工具来更新安全编码规则包。该选项专为那些在安装期间无法访问 Internet 站点的用户提供。

## 8.2.3　安装 Fortify SCA

### 1. 安装选项

Windows MSI 安装程序可安装以下任意(及所有)组件：Fortify SCA sourceanalyzer、AuditWorkbench、用于基于 Eclipse 的 IDE 的安全编码插件、用于 Visual Studio 的安全编码包以及 Fortify Team Server Web 应用程序。

Linux TAR 文件可安装以下组件：

- Fortify SCA
- Sourceanalyzer
- Fortify SCA 套件：
- Sourceanalyzer
- Audit Workbench
- Eclipse 安全编码插件

注意　用于安装 Eclipse 3+安全插件的选项可安装所有基于 Eclipse 的 IDE 插件，但用于 IBM WebSphere Studio Application Developer 5 的插件除外。

- Fortify Team Server

- Team Server Web 应用程序

**注意** 如果使用的 Linux 平台正在运行安全增强版的 kernel (SELinux)，可能无法完成安装进程，除非禁用此项服务。安装结束后，可重新启用该服务。

Mac OS X DMG 文件可安装以下组件：

- Fortify SCA
- Sourceanalyzer
- Fortify SCA 套件：
- Sourceanalyzer
- Audit Workbench
- 用于基于 Eclipse 的 IDE 的源代码插件

**注意** 用于安装 Eclipse 3+ 安全插件的选项可安装所有基于 Eclipse 的 IDE 插件，但用于 IBM WebSphere Studio Application Developer 5 的插件除外。

- Fortify Team Server
- Team Server Web 应用程序

Unix TAR 文件可安装以下组件：

- Fortify SCA
- Sourceanalyzer
- Fortify Team Server
- Team Server Web 应用程序

### 2. 在 Windows 上安装

在 Windows 上，使用 InstallShield 向导应用程序来引导安装。

安装 Fortify SCA：

1) 导航至包含下载软件的目录，如图 8-1 所示。

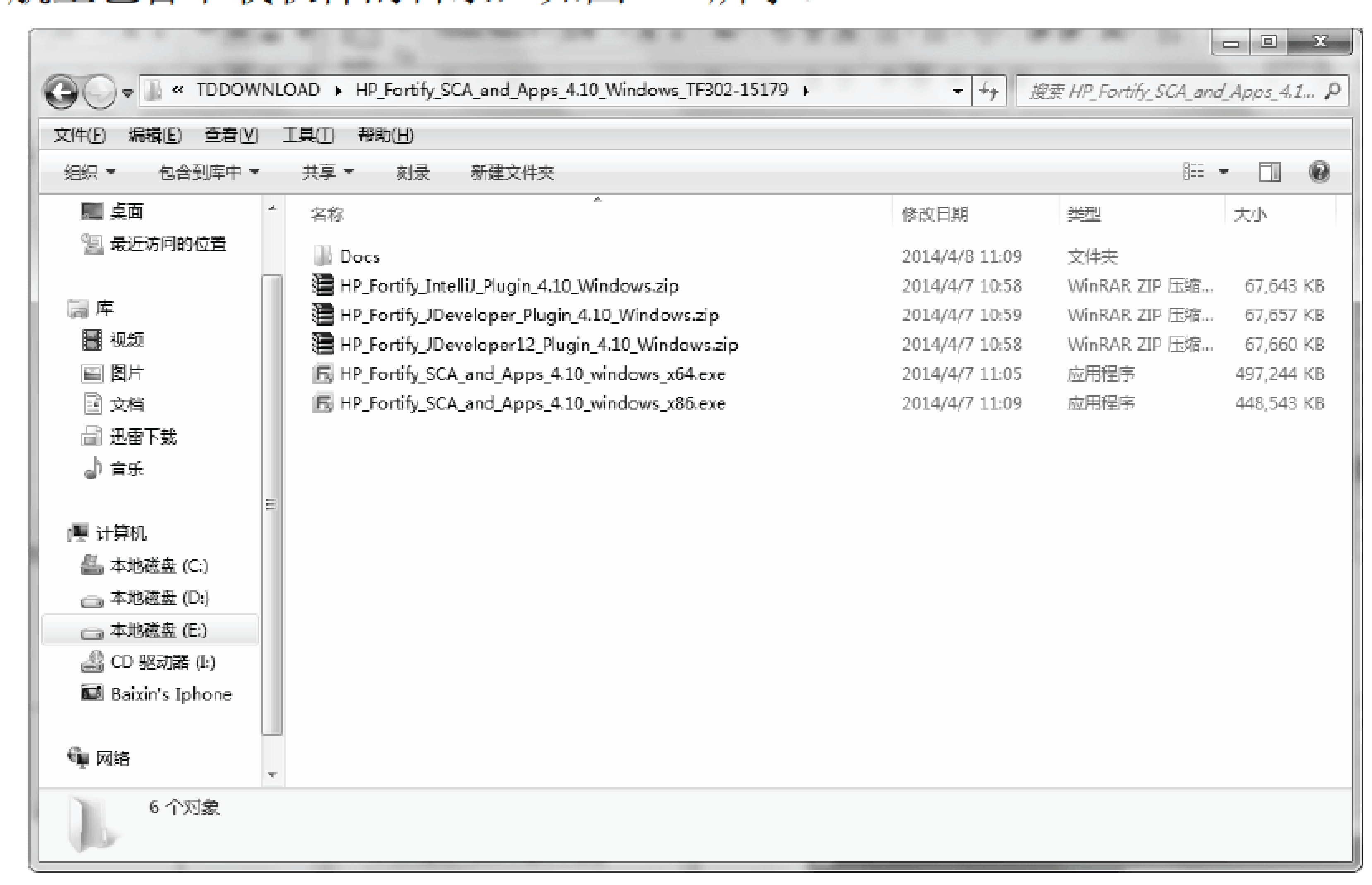



图 8-1　导航目录

2) 双击 install.exe，如图 8-2 所示。

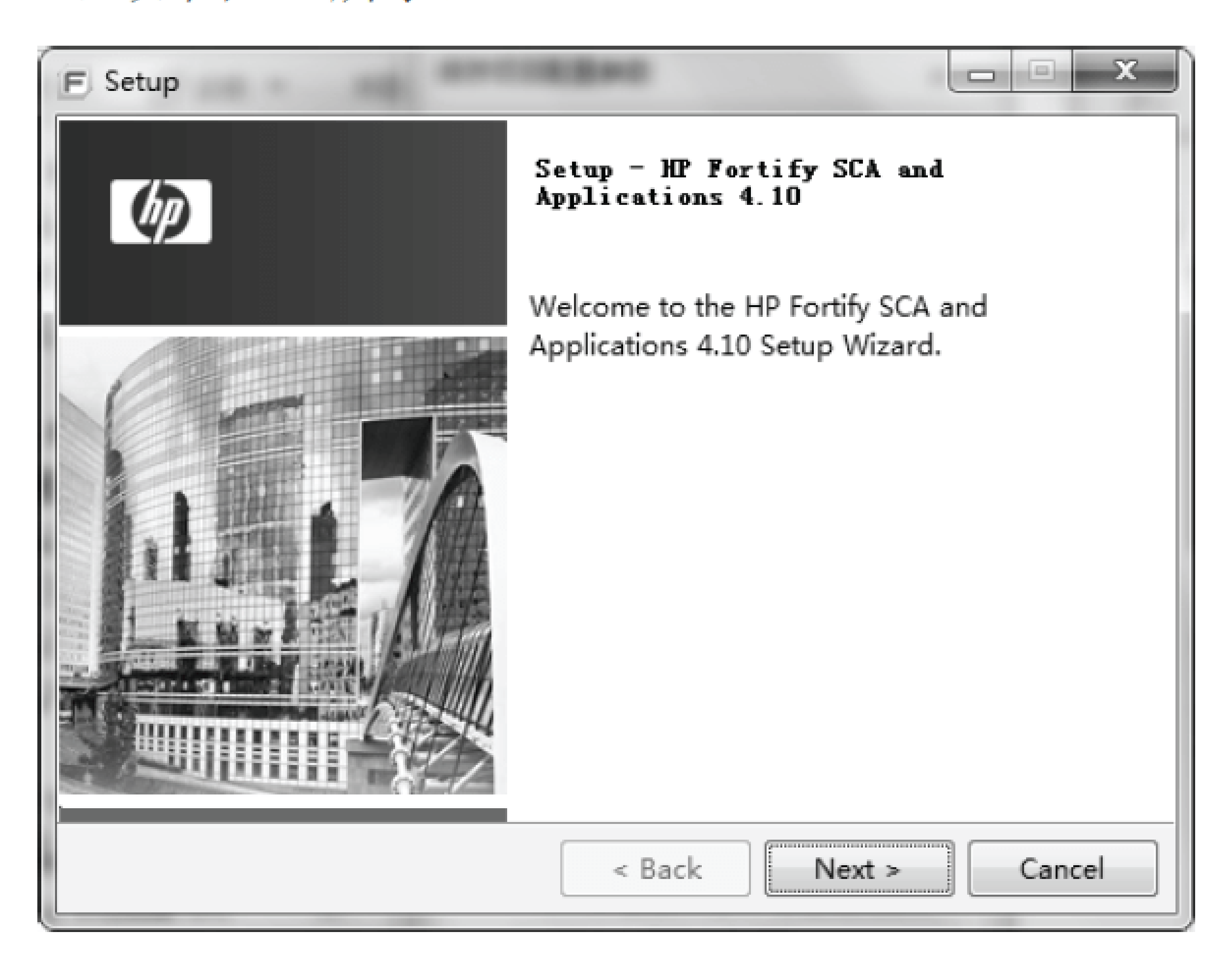



图 8-2　install.exe 启动界面

3) 单击 Next 并接受许可证协议，如图 8-3 所示。




图 8-3　协议许可证协议界面

4) 选择安装路径，如图 8-4 所示。

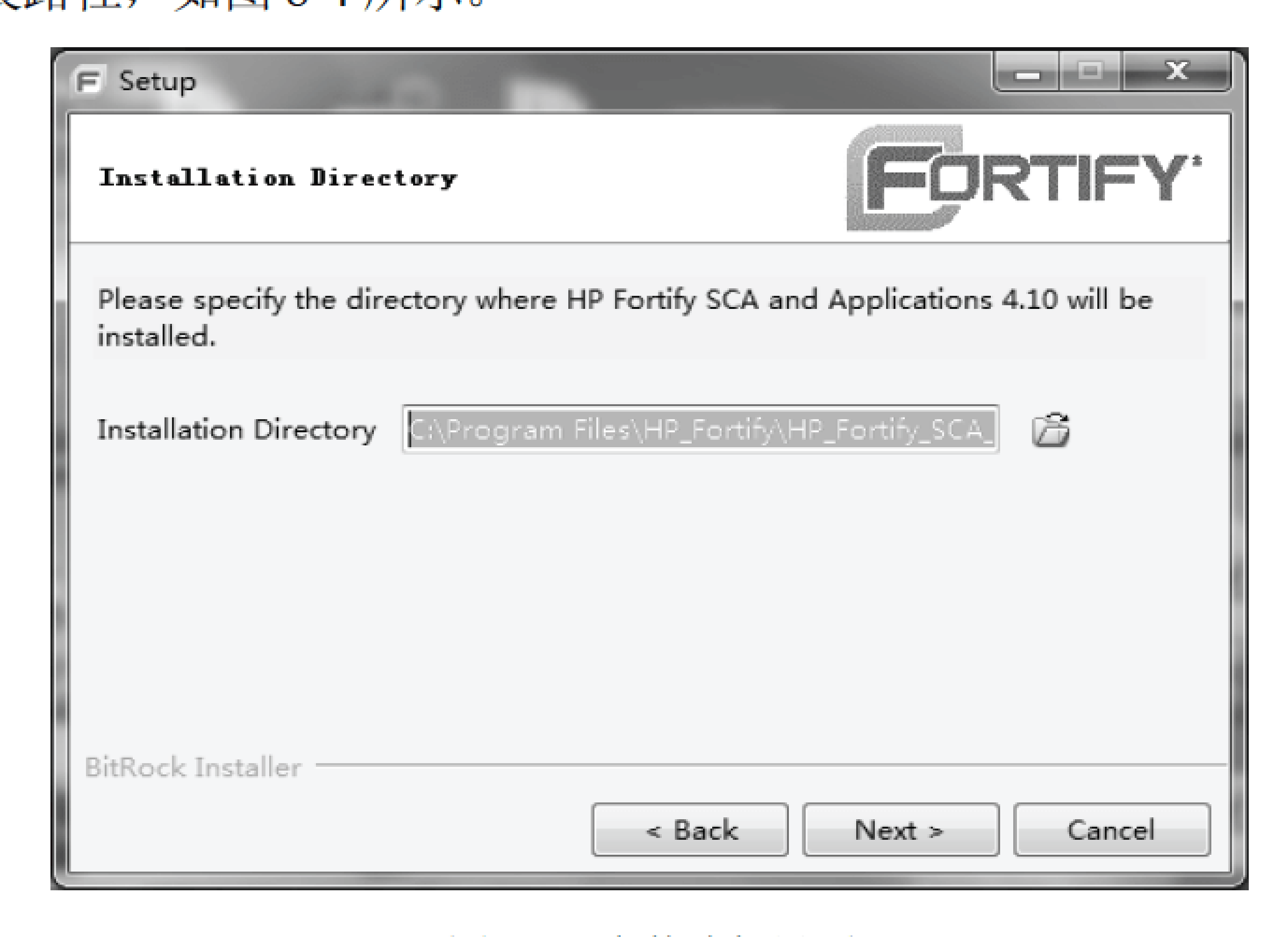



图 8-4　安装路径界面

5) 选择要安装的组件，如图 8-5 所示。

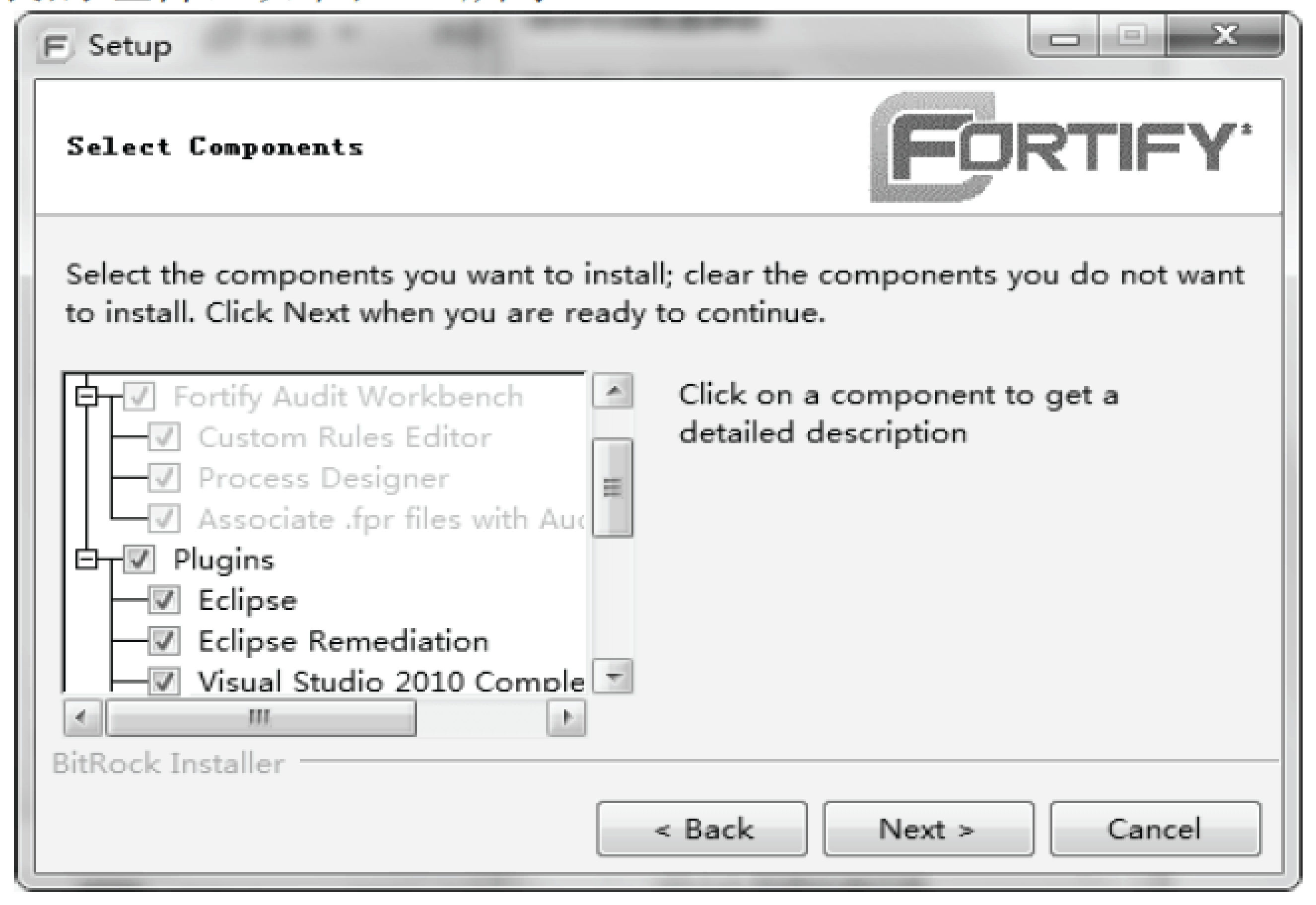



图 8-5　选择安装组件

6) 选择安装许可证位置，如图 8-6 所示。

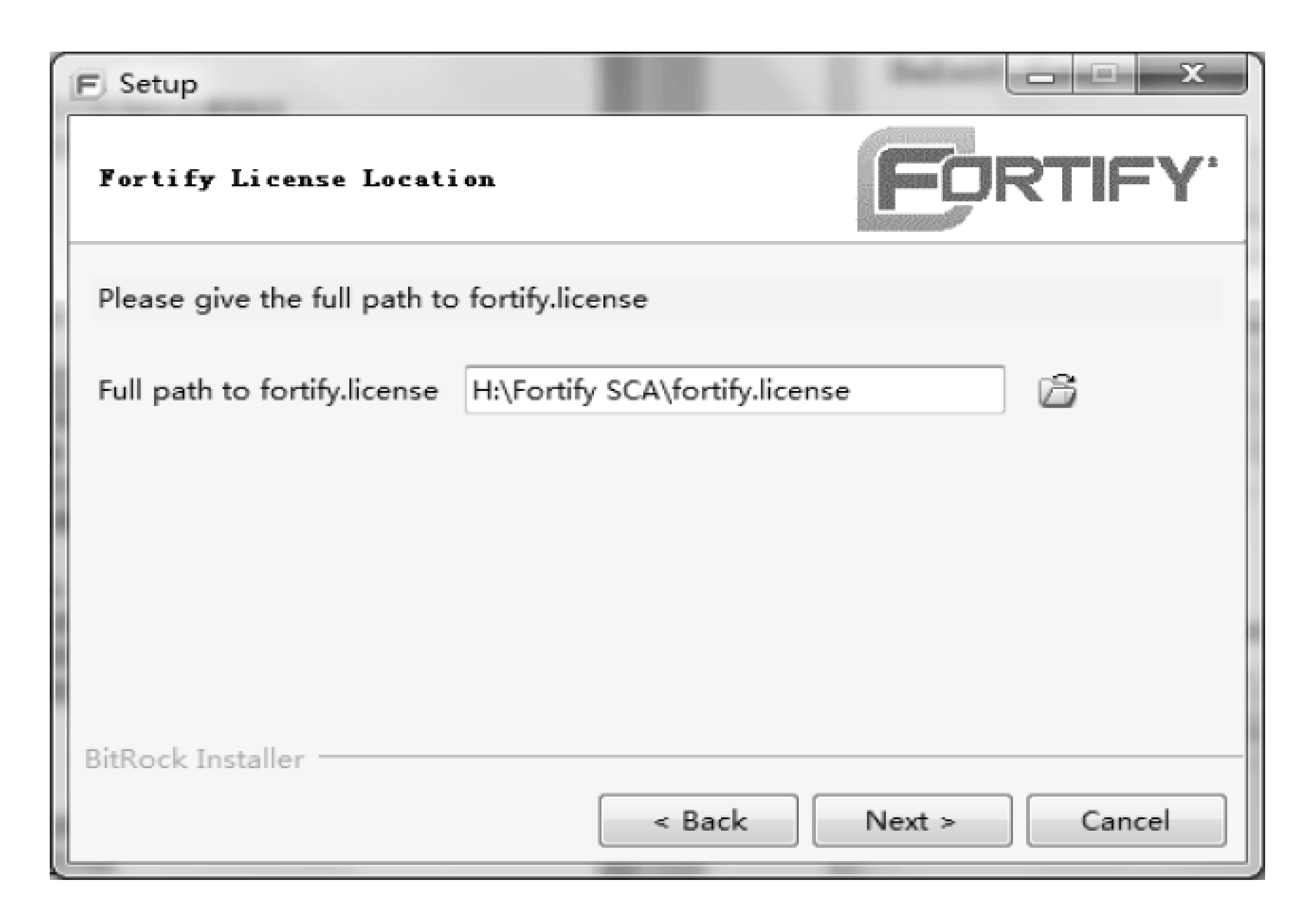



图 8-6　选择许可证路径

7) 选择 Rulepack 位置，单击 Next，如图 8-7 所示。




图 8-7　选择 Rulepack 位置

8) 确定 SCA 移除选项，点击 Next，如图 8-8 所示。

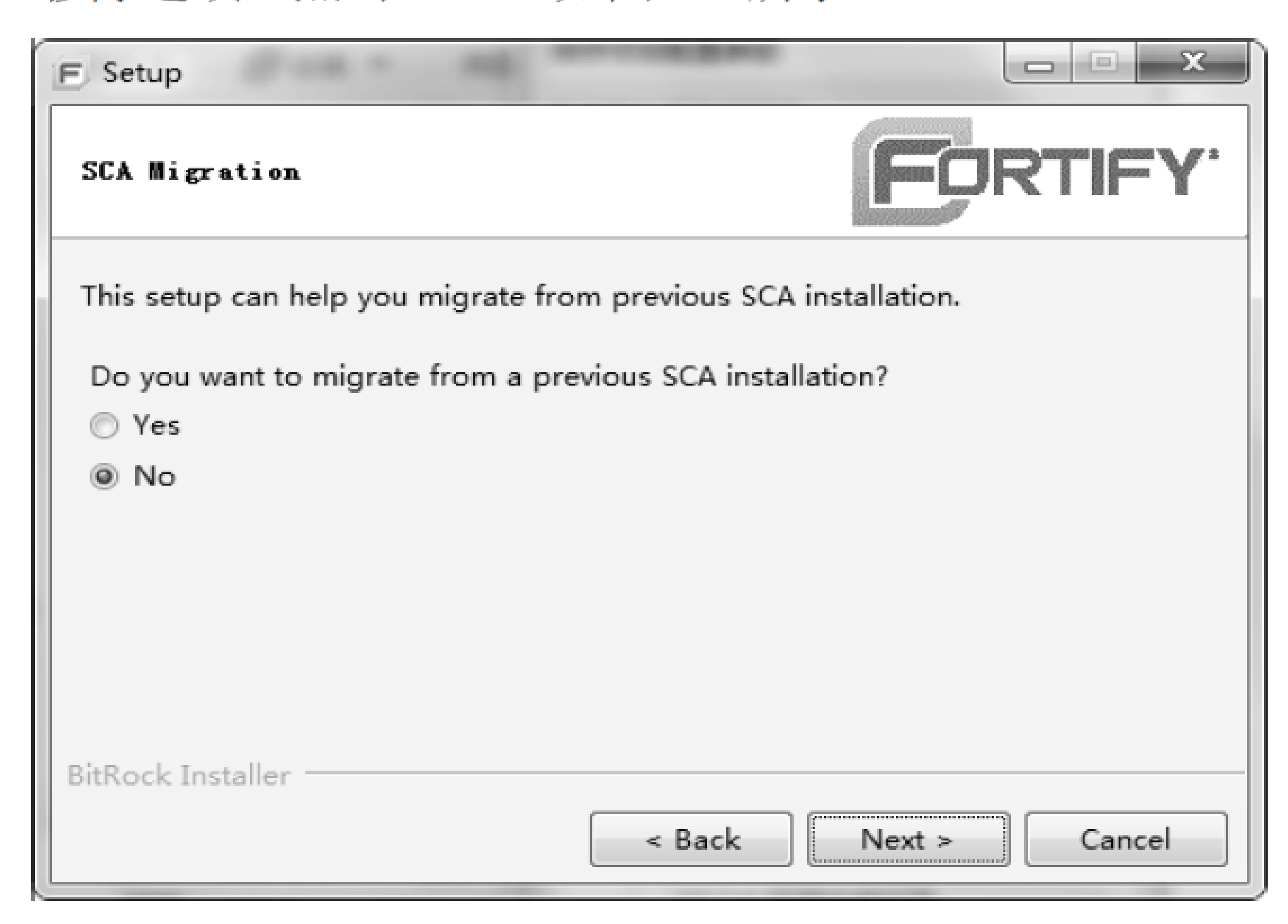



图 8-8　选择 SCA 移除

9) 准备安装，如图 8-9 所示。

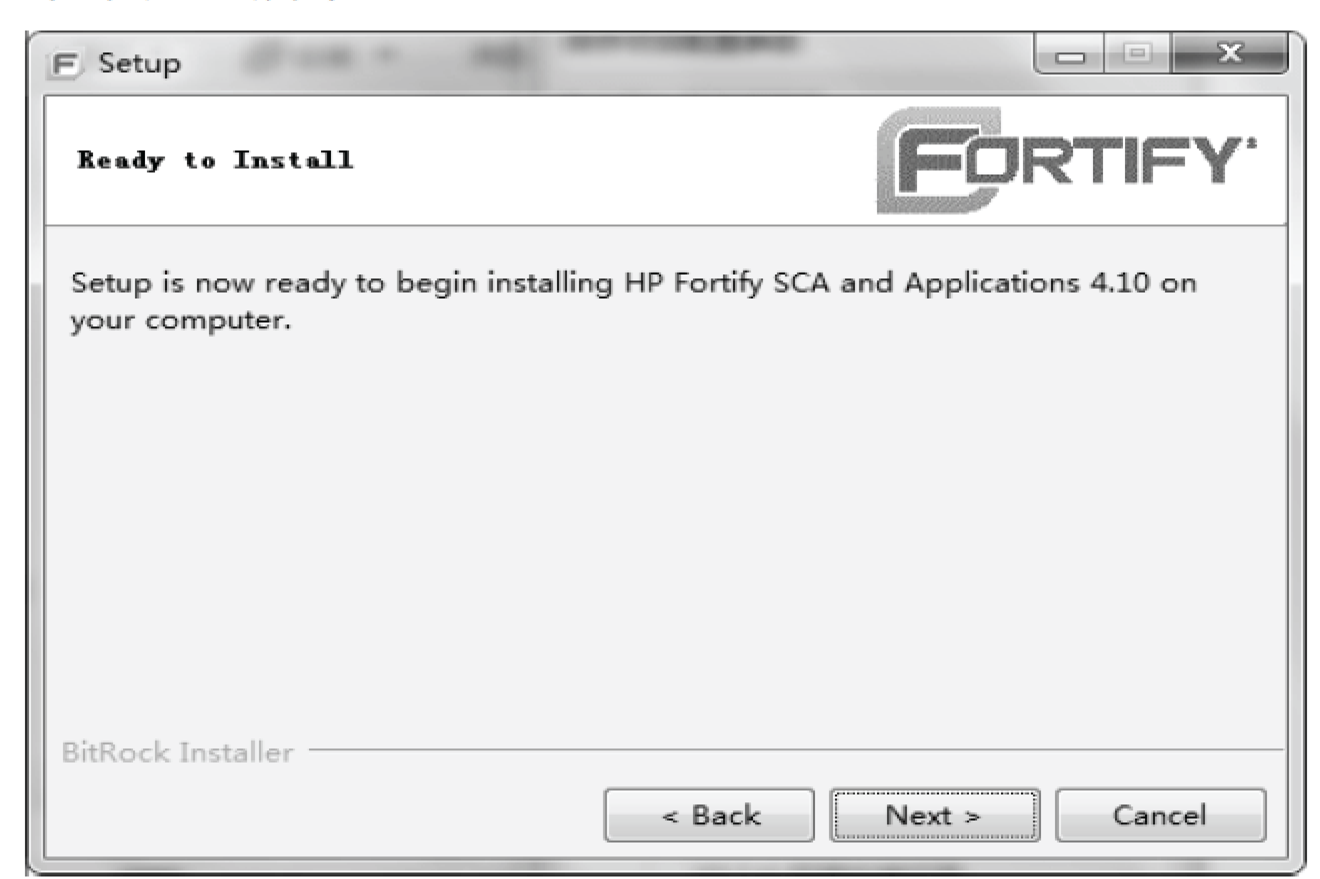



图 8-9　准备安装界面

10) 安装完成，如图 8-10 所示。




图 8-10　安装完成界面

11) 核实下载的 Rulepacks，如图 8-11 所示。




图 8-11　核实下载 Rulepacks

12) 执行“sourceanalyzer –v”验证安装，如图 8-12 所示。

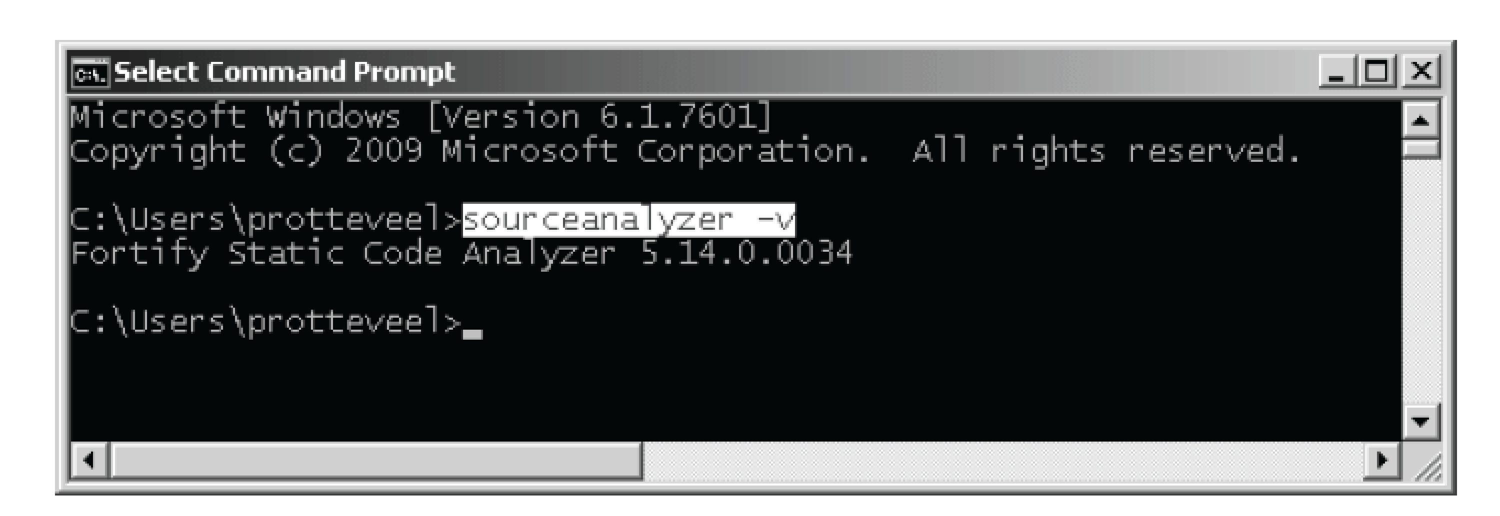



图 8-12　验证安装

13) HP Fortify 会自动加入到 MS 或 VS 中，如图 8-13 所示。

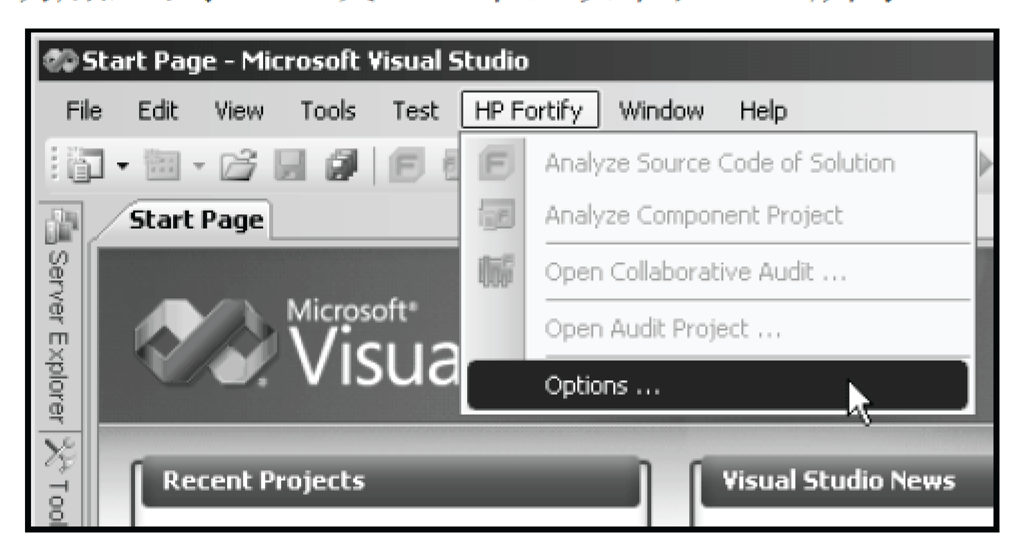



图 8-13　MS 中的 HP Fortify 插件

14) 安装 Eclipse，选择 Help/Install New Software，如图 8-14 所示。

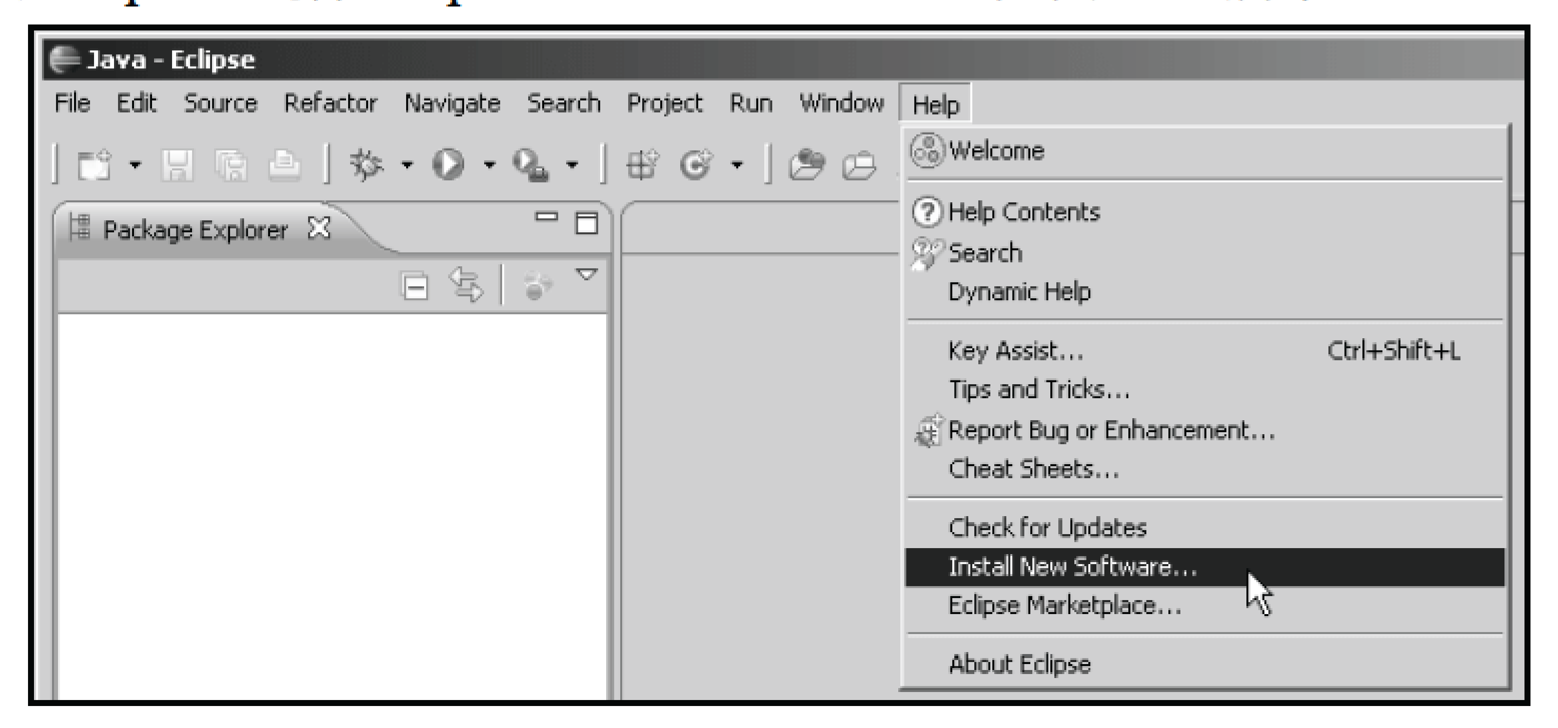



图 8-14　选择安装新软件

15) 选择<Fortify install>/plugins/eclipse 并单击 OK，如图 8-15 所示。

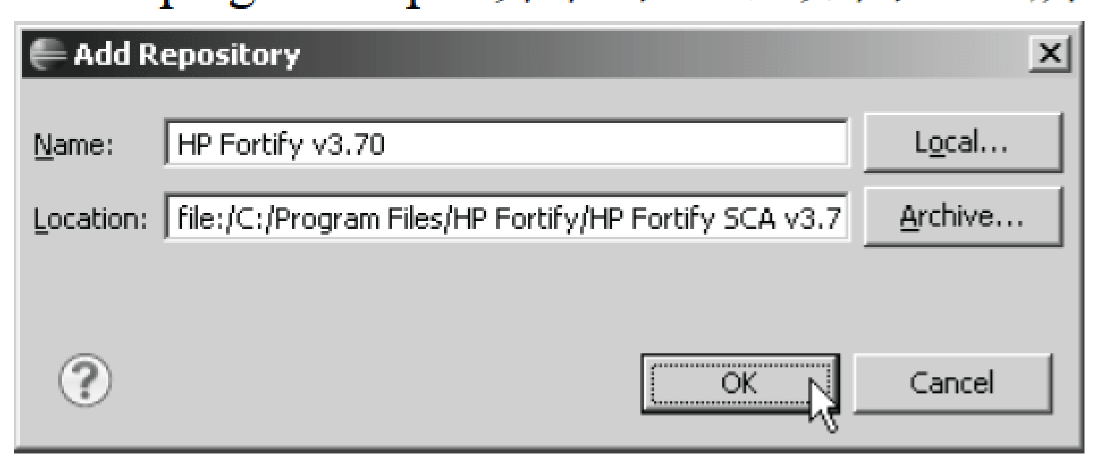



图 8-15　安装插件路径界面

16) 选择组件并单击 Next，如图 8-16 所示。

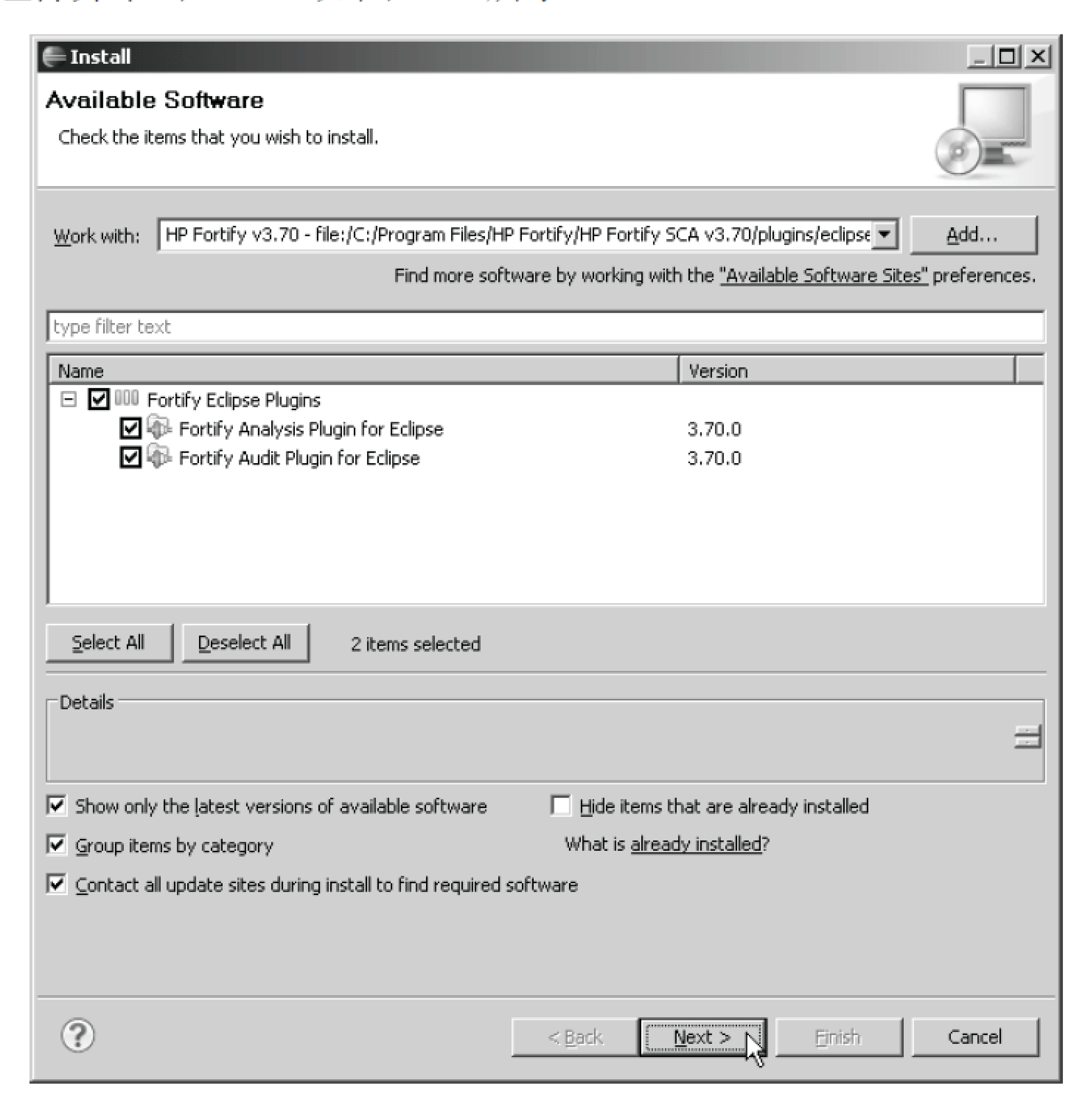



图 8-16　选择组件界面

17) 确定安装组件并单击 Next，如图 8-17 所示。

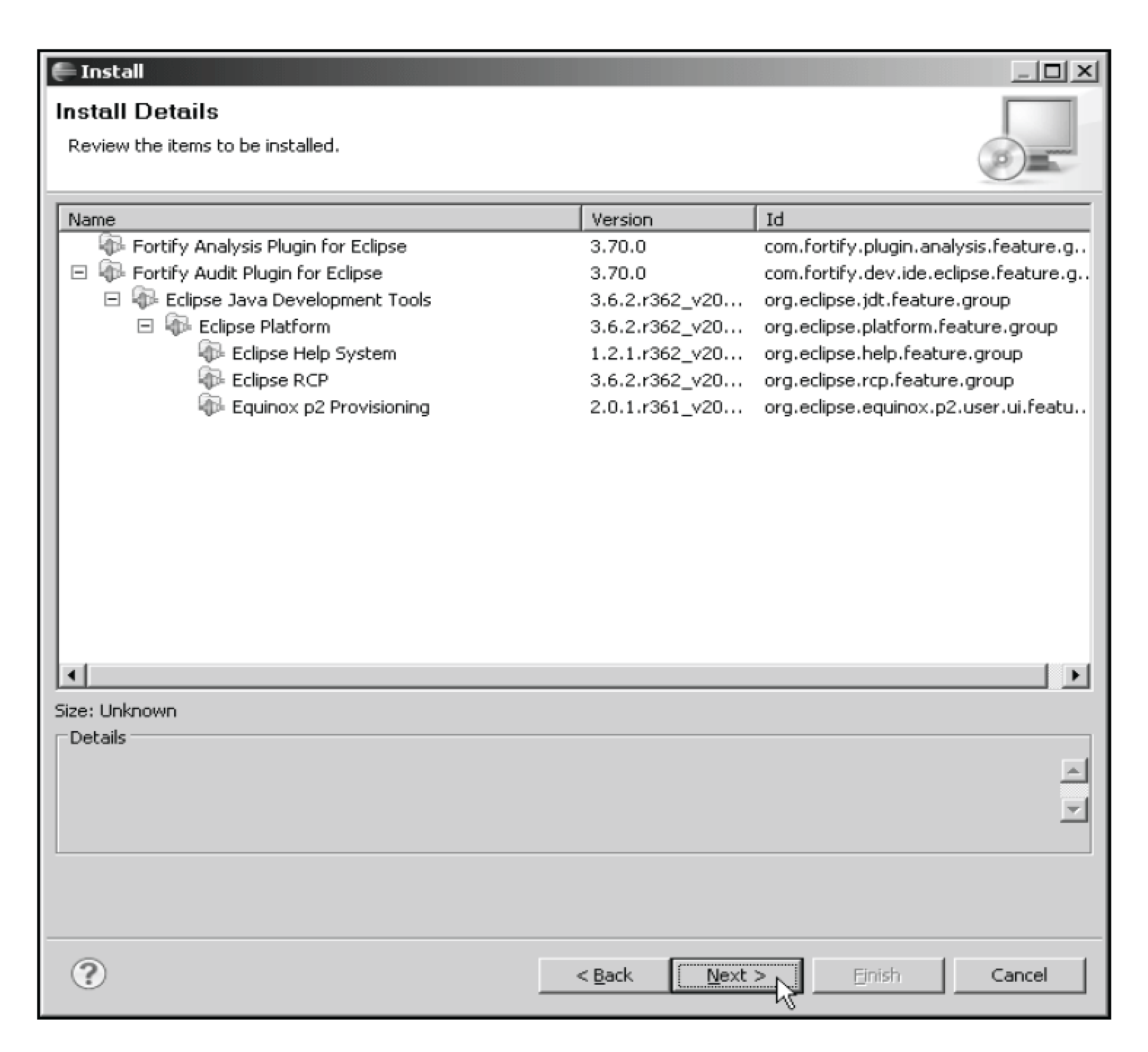



图 8-17　确定组件界面

18) 阅读并接受许可证，单击 Next，如图 8-18 所示。

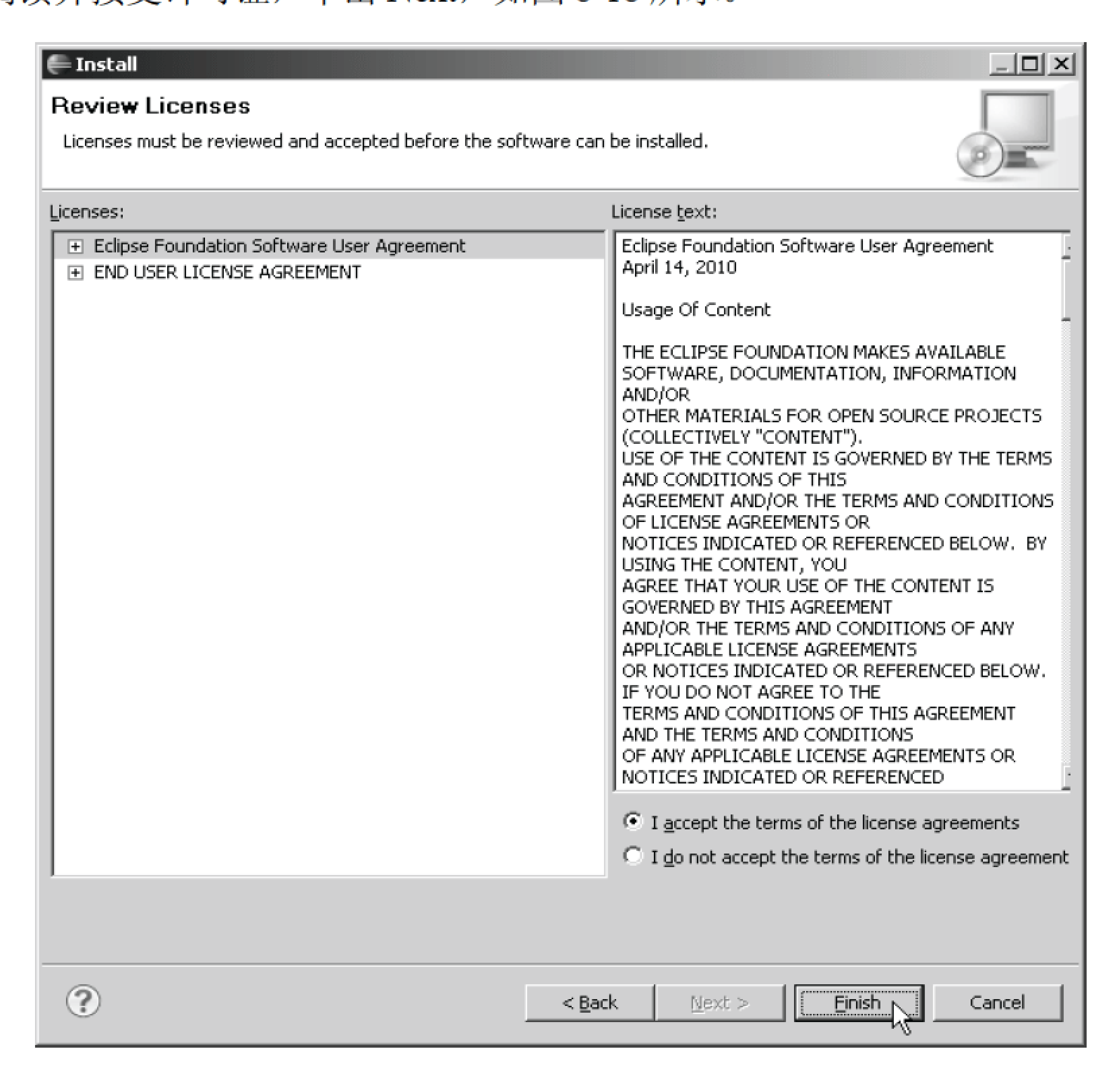



图 8-18　阅读接受许可证界面

19) 重启 Eclipse，如图 8-19 所示。




图 8-19　重启 Eclipse

20) HP Fortify 已被加入到 Eclipse，如图 8-20 所示。

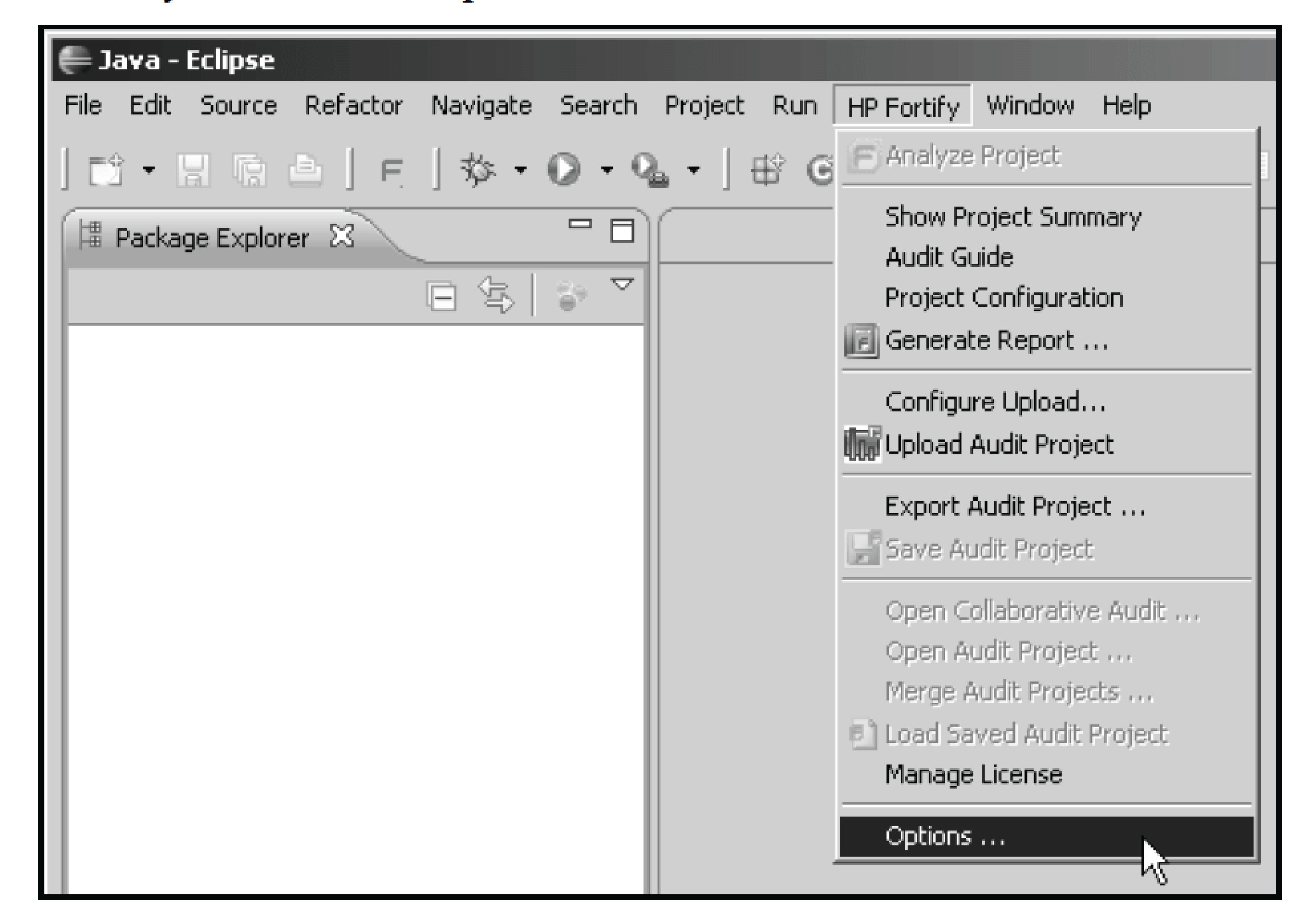



图 8-20　安装完成插件

### 3. 在 Linux 上进行安装

安装 Fortify SCA：

1) 导航至包含下载软件的目录。

2) 打开一个 shell 窗口，并通过提示输入以下命令来提取文件：

```
gunzip<archive_name>.tgz
tar -xvf <archive_name>.tar
```

3) 有关以下主题的信息，请参见“后续安装任务”：

- 从之前版本中迁移属性文件
- 为代理规则包更新服务器指定连接信息
- 指定一种非英文的语言环境
- 运行 Rulepack Updater

### 4. 在 Unix 上进行安装

安装 Fortify SCA：

1) 导航至包含下载的软件的目录。

2) 打开一个 shell 窗口并通过输入以下命令来提取文件：

```
gunzip<archive_name>.tgz
gtar -xvf <archive_name>.tar
```

3) 有关以下主题的信息，请参见“后续安装任务”：

- 从之前版本中迁移属性文件；
- 为代理规则包更新服务器指定连接信息；
- 指定一种非英文语言环境；
- 运行 Rulepack Updater。

### 5. 在 Mac OS X 上进行安装

安装 Fortify SCA：

1) 导航至包含下载软件的目录。

2) 双击 DMG 文件。

系统会创建一个虚拟驱动器，其中包含产品文件夹。

3) 将文件夹复制并粘贴至选择的安装位置，如下例所示：

```
/Applications/Fortify Software/...
```

4) 有关以下主题的信息，请参见“后续安装任务”：

- 从之前版本中迁移属性文件；
- 为代理规则包更新服务器指定连接信息；
- 指定一种非英文语言环境；
- 运行 Rulepack Updater。

注意：要在 Mac OS X 中启动 Audit Workbench，请在终端窗口中输入以下命令：

```
# cd /Applications/Fortify Software/Fortify-SCA-5.0/bin
# ./auditworkbench
```

### 6. 安装 Eclipse 安全编码插件

安装基于 Eclipse 的 IDE：

1) 按照上述说明安装 Fortify SCA 套件。

注意：对于 Windows 平台来说，请确保在安装期间选择了 Eclipse 3.x 选项。

2) 打开 Eclipse。

3) 选择 Help（帮助），再依次选择 Software Updates（软件更新）和 Manage Configuration Location

(管理配置位置)。

4) 单击 Add an Extension Location（添加扩展位置）。

5) 选择<install_directory>/plugins/eclipse。

6) 单击 OK (确定)。

屏幕上将显示 Fortify Software 安全编码插件菜单。

## 8.2.4　后续安装任务

本节将介绍如何升级安全编码规则包，如何在升级之后迁移属性文件、指定不同的语言环境、为规则包更新服务器指定不同的 URL 以及为规则包更新服务器指定一个代理服务器。

升级规则包在“更新规则包”中有所描述。余下的任务是使用后续安装工具，相关内容在“运行后续安装工具”中有所描述。

### 1. 运行后续安装工具：

1) 在命令行中，导航至 bin 目录。
2) 输入 scapostinstall 以启动该工具。
3) 选择所需选项并输入相关值以执行以下任务。

- 迁移属性文件；
- 指定语言环境；
- 为 Rulepack Updates 指定一个代理服务器。

4) 输入 s 显示设置，输入 r 返回前面的提示，输入 q 退出该工具。

### 2. 迁移属性文件

将早期版本中更改过的属性文件迁移到当前版本：

启动后续安装工具。

1） 输入 1 以选择 Migration。
2） 输入 1 以选择 SCA Migration。
3） 输入 1 以选择 Set previous migration directory。
4) 输入之前的安装目录
5) 输入 s 确认设置。
6) 输入 2 以执行迁移。
7) 输入 y 进行确认。

### 3. 指定语言环境

默认情况下，Fortify Source Code Analysis 的安装环境为英文。

指定一个不同的环境：

1) 启动后续安装工具。
2) 输入 2 以选择 Settings。
3) 输入 1 以选择 General。
4) 输入 1 以选择 Locale。
5) 输入环境代码：

- 英文：en
- 日文：ja
- 韩文：ko

- 简体中文：zh_CN
- 繁体中文：zh_TW

### 4. 为 Rulepack Updates 指定一个代理服务器

为 rulepack update 服务器指定一个代理：

1) 启动后续安装工具。
2) 输入 2 以选择 Settings。
3) 输入 2 以选择 Rulepack Update...。
4) 输入 2 以选择 Proxy Server Host。
5) 输入代理服务器的名称。
6) 输入 3 以选择 Proxy Server Port。
7) 输入代理服务器的端口号。

更新规则包使用 Rulepack Update 工具，从远程服务器或本地下载的文件中更新规则包。有关下载规则包的信息，请参见“下载软件”。

### 5. 更新规则包

1) 在命令行中，导航至 bin 目录。
2) 输入 rulepackupdate 以启动该工具。

系统会响应操作，可能会显示一个错误消息，也可能显示一个已下载的规则包列表。

注意：本版本不支持从 Fortify Manager 中下载规则包。

如果具有之前从 Fortify Customer Portal 下载的规则包，请运行带有-import 的 rulepackupdate 选项，以及已下载的规则包的目录路径。

## 8.2.5 卸载 Fortify SCA

### 1. 在 Windows 平台上进行卸载

要卸载 Windows 上的 Fortify SCA 软件，请使用“控制面板”上的“Windows 添加或删除程序”实用程序。

1) 选择开始→控制面板→添加或删除程序→Fortify SCA 5.0。
2) 在 Fortify SCA 5.0 下，单击 Delete(删除)。

### 2. 在其他平台上进行卸载

在 Mac OS X、Linux 和 Unix 平台上卸载 Fortify SCA 软件：

1) 备份配置，包括创建的所有重要文件。
2) 使用以下命令手动删除安装目录：

```
rm -rf <install_directory>/FortifySCA
```

# 8.3　静态代码分析器分析原理

## 8.3.1　分析阶段概述

源代码分析过程包含下列几个阶段：

- 构建集成：第一阶段包括决定是否将 SCA 集成到构建编译程序。
- 转换：通过一系列命令聚集起来的源代码会被转换为与一个构建 ID 相关联的中间格式，该构建 ID 通常就是正在扫描的项目的名称。
- 分析：系统将扫描在转换阶段识别出的源文件，随后生成一个分析结果文件，该文件通常为 Fortify 项目(FPR)格式。FPR 文件通过.fpr 文件扩展名来表示。
- 对转换和分析阶段的验证：确保源文件的扫描使用了正确的规则包，而且没有报告重大错误。

## 8.3.2　分析命令示例

以下示例用来分析代码的命令序列：

>sourceanalyzer -b <build_id> -clean

>sourceanalyzer -b <build_id> ...

>sourceanalyzer -b <build_id> -scan -f results.fpr

为同时分析多个构建，添加额外的构建作为参数:

sourceanalyzer -b <build_id1> -b <build_id2> -b <build_id3> -scan -f results.fpr

## 8.3.3　内存注意事项

SCA 运行时，所需的物理 RAM 取决于许多因素。因为每个客户情况都是不同的, 一些因素(包括源文件的大小和复杂性)，使其无法量化。如果你遇到一个低内存错误，增加 SCA 可用内存可能会解决问题。

默认情况下，Fortify SCA 最多可使用 600 MB 的内存。如果该内存仍不能满足分析某个特定代码库的需要，可能需要在扫描阶段提供更多的内存。这可以通过在 sourceanalyzer 命令中使用-Xmx 选项来实现。例如，要让 Fortify SCA 可用的内存达到 1000 MB ，可在命令中包含-Xmx1000M 选项。

注意, 为 Fortify SCA 分配的内存不应高于计算机本身的可用内存,因为这样会降低性能。指导原则是在没有运行其他内存密集型程序的情况下，为其分配的内存不能超过 2/3 的可用物理内存。

# 8.4　静态代码分析器分析过程

图 8-21 显示了分析过程。

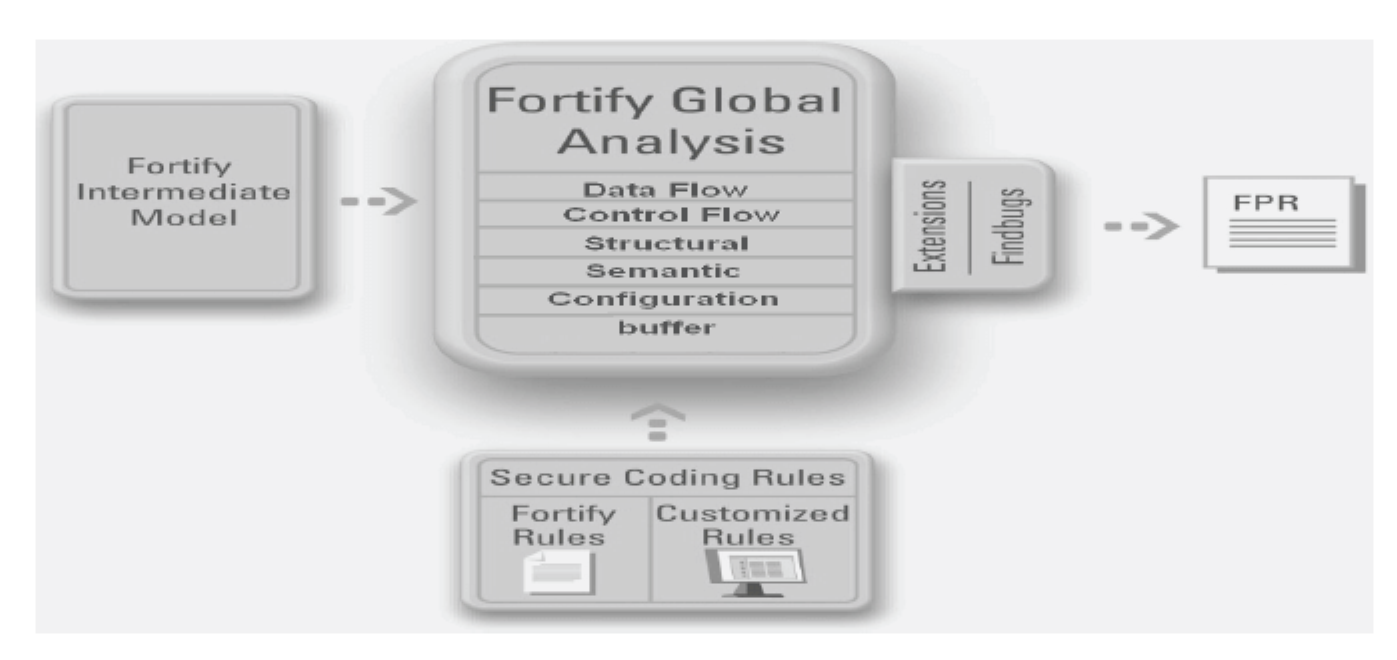



图 8-21　分析过程

静态代码分析器分析过程是由分析器和规则组成的。

1) 分析器

- 包括 6 个独立的分析器(分析器是静态分析方法);
- 每个分析器由相应的规则类型相互联系起来。

2) 规则

- 对于易损点的定义;
- 分析器有很多的规则类型(每个规则类型匹配一个分析器，并且有其独立的语法);
- 安全编码规则包(一系列包含规则的文件)。

HP Fortify SCA 包括 6 个主要分析器，下面介绍主要分析器的作用和简单示例。

1) 数据流主要作用(如图 8-22 所示):

- 查找漏洞，其中不受信任的输入可能会控制应用程序的操作;
- 分析器采用全球漏洞传播技术追踪非信任的数据流。

```
import java.sql.*;

public class SQLInjection {
public static void main (String args[]) {
   Connection conn = null;
   try {
      String userName = args [0];
      String passwd = args [1];
      String query = "select uname, passwd from users where uname like "+userName+"%";
      conn = DriverManager.getConnection ("jdbc:odbc:logistics", "admin", "letmein");
      Statement stmnt = conn.createStatement ();
      ResultSet rs = stmnt.executeQuery (query);
      while ( rs.next() ) {
         ...
         }
      rs.close ();
      stmnt.close ();
      conn.close ();
      }
   catch (SQLException err) {
     err.printStackTrace ();
     }
   }
}
```

Source

args[] is tainted

Taint propagated to userName

Sink

Vulnerability Reported

Tainted input userName used in assignment

图 8-22　数据流示例

2) 控制流主要作用(如图 8-23 所示):

```
private void cmdLogin_Click(object sender, System.EventArgs e){
    Label lblMsg = null;
    try {
        string txtUser = Request.Form["txtUsername"];
        string txtPassword = Request.Form["txtPassword"];
        string strCnx = "server=localhost;database=northwind;uid=sa;pwd=;";
        SqlConnection cnx = new SqlConnection(strCnx);
        cnx.Open();
        string strQry = "SELECT Count(*) FROM Users WHERE UserName='" +
        txtUser + "' AND Password='" + txtPassword + "'";
        int intRecs;

        SqlCommand cmd = new SqlCommand(strQry, cnx);
        intRecs = (int) cmd.ExecuteScalar();

        if (intRecs > 0){
            FormsAuthentication.RedirectFromLoginPage(txtUser, false);
            cnx.Close();
        }
        else{
            lblMsg.Text = "Login attempt failed.";
        }
    }
    catch (Exception err){
        Response.Write(err.StackTrace.ToString());
    }
}
```



图 8-23　控制流示例

- 检测操作/函数调用存在潜在危险的序列；
- 控制流分析器通过源代码(NST 文件)确定状态转换来模拟状态机；
- 控制流规则是状态机的定义。

# 8.5　静态代码分析器扫描的方式

1) HP Fortify 扫描向导

HP Fortify 扫描向导是一个实用程序，它可以快速轻松地准备和扫描 SCA 项目代码。扫描向导能够运行本地扫描，如果正在使用 HP Fortify 云扫描，在处理器密集扫描分析阶段，云计算机可以相互分担压力。

2) HP Fortify 云扫描

HP Fortify 云扫描可以帮助使用 HP Fortify 静态代码分析器的用户更好地管理资源，在处理器密集扫描分析阶段，可从一台构建计算机卸载处理器，转移到准备好的云计算机上。

构建机器上转换阶段完成后，生成 SCA 移动建立会话，云扫描将它转移到可用的计算机进行扫描。除了释放构建的机器，这个过程根据需要添加更多的资源到云端，从而不必中断构建过程，使它更容易扩展系统。

此外，软件安全中心的用户可以直接使用云扫描将 FPR 文件输出到服务器。

# 8.6　静态代码分析器转换源代码

## 8.6.1　转换阶段

用来执行第一个分析阶段(即文件转换)的基本命令行语法是：

sourceanalyzer -b <build_id> ...

转换阶段包含一次或多次使用 sourceanalyzer 命令调用 Fortify SCA 的过程。构建 ID(-b <build-id>) 用于将这些调用过程绑定到一起。

随后调用 sourceanalyzer 时会将任何新指定的源文件或配置文件添加到与构建 ID 相关联的文件列表中。转换结束时，可使用-show-build-warnings 指令列出在转换过程中遇到的所有警告和错误：

sourceanalyzer -b <build_id> -show-build-warnings

要查看与某个特定构建 ID 相关联的所有文件，可使用-show-files 指令：

sourceanalyzer -b <build_id> -show-files

下面将介绍如何转换不同类型的源代码：

- 转换 Java 源代码
- 转换.NET 源代码
- 转换.C 和 C++ 代码
- 转换 ABAP/4
- 转换 FLEX
- 转换移动平台代码
- 转换其他语言，如 ColdFusion、Classic ASP 以及 JavaScript

1) SCA 移动建立会话(可选)

SCA 移动建立会话允许一个项目在一台计算机上翻译并在另一台计算机上进行分析。当创建一个 SCA 移动建立会话时，生成一个.mbs 包括会话目录中分析阶段所需文件。然后将.mbs 文件被转换到不同机器上进行分析。

2) 如何创建 SCA 移动建立会话

在转义完成的那台计算机上，发出以下命令来生成一个 SCA 移动建立会话:

sourceanalyzer -b <build_id> -export-build-session <file.mbs>

其中<file.mbs>是分配给 SCA 移动建立会话的文件名。

3) 如何引入 SCA 移动建立会话

一旦把.mbs 文件移动到你想运行分析的机器上，发出以下命令：

sourceanalyzer -import-build-session <file.mbs>

其中<file.mbs>是 SCA 移动建立会话名称。

一旦你已经导入 SCA 移动建立会话，你就准备继续进行分析阶段。

4) 分析阶段

本主题将介绍分析阶段所用的语法：扫描在转换过程中创建的中间文件并创建分析结果文件。分析阶段包含对源分析器的一次调用，需要指定构建 ID，并包含 -scan 指令以及任何必要的分析或输出选项。

注意：默认情况下， Fortify SCA 会包含 FPR 中的源代码。

分析阶段所用的基本命令行语法为：

```
sourceanalyzer -b <build_id> -scan -f results.fpr
```

为一次运行分析多个构建，添加额外的构建命令行：

```
sourceanalyzer - b <build_id1> -b <build_id2> -b <build_id3> -scan -f results.fpr
```

为一次运行分析多个无记录构建，添加额外的构建命令行：

```
sourceanalyzer -b <build-id1> -b <build-id2> -b <build-id3> -auth-silent -scan -f results.fpr
```

5) 对转换和分析阶段的验证

Audit Workbench 的“结果认证”功能可用于检验分析是否完成。“结果认证”可显示关于 FortifySCA 扫描过的代码的具体信息，包括：

- 扫描的文件列表，包括文件大小和时间戳；
- 用于转换的 Java 类路径；
- 分析使用的规则包列表；
- Fortify SCA 运行时设置和命令行参数的列表；
- 在转换和分析过程中遇到的问题或者警告的列表；
- 机器/平台信息。

## 8.6.2 转换 Java 源代码

### 1. Java 命令行语法

基本的 Java 命令行语法是：

```
sourceanalyzer -b <build_id> -cp <classpath><file_list>
```

对于 Java 代码，Fortify SCA 既可模拟编译器(这可能有利于进行构建集成)，也可直接接受源文件(这有利于进行命令行扫描)。

要让 Fortify SCA 模拟编译器，请输入：

```
sourceanalyzer -b <build_id> javac [<compiler options>]
```

要直接向 Fortify SCA 传文件，请输入：

```
sourceanalyzer -b <build_id> -cp <classpath> [<compiler options>] \
<files>|<file specifiers>
```

其中：

<compiler options>是传到编译器的选项。

-cp <classpath>指定用于 Java 源代码的类路径。类路径是一个构建目录和 jar 文件的列表。格式与 javac 正常使用的格式相同(使用冒号或分号分隔的路径列表)。可使用 Fortify SCA 文

件说明符。

-cp "build/classes:lib/*.jar"

注意：如果未使用此选项指定类路径，系统将使用 CLASSPATH 环境变量。

### 2. Java 命令行示例

要在类路径上使用 j2ee.jar 转换单个名为 MyServlet.java 的文件，请输入：

```
sourceanalyzer -b MyServlet -cp lib/j2ee.jar MyServlet.java
```

要使用 lib 目录中的所有 jar 文件作为类路径来转换 src 目录中的所有.java 文件，请输入：

```
sourceanalyzer -b MyProject -cp "lib/*.jar" "src/**/*.java"
```

要在运行 javac 编译器的同时转换 MyCode.java 文件，请输入：

```
sourceanalyzer -b mybuild javac -classpath libs.jar MyCode.java
```

### 3. 使用 Fortify Ant Compiler Adapter 与 Ant 集成

Fortify SCA 提供了一个 Ant Compiler Adapter，如果项目使用 Ant 构建文件，则可以用它轻松转换 Java 源文件。这种集成只需要设置两个 Ant 属性，并且不必修改 Ant build.xml 文件，通过命令行即可完成。执行构建时，Fortify SCA 会拦截所有的 javac 任务调用，并在 Java 源文件编译完成后对其进行转换。

请注意，任何 JSP 文件、配置文件或应用程序中包含的其他非 Java 源文件都需要在一个单独的步骤中进行转换。

要使用 Compiler Adapter，需执行以下步骤：

- sourceanalyzer 可执行文件必须在系统 PATH 上；
- sourceanalyzer.jar (位于 Core/lib 中)必须在 Ant 的类路径上；
- build.compiler 属性必须设置为 com.fortify.dev.ant.SCACompiler；
- sourceanalyzer.buildid 属性必须设置为构建 ID。

以下示例演示了如何在不修改构建文件的情况下使用 Compiler Adapter 运行 Ant 构建：

```
ant -Dbuild.compiler=com.fortify.dev.ant.SCACompiler \
-Dsourceanalyzer.buildid=MyBuild \
-lib <install_dir>/Core/lib/sourceanalyzer.jar
```

-lib 选项只适用于 Ant 1.6 或更高版本。在较低版本中，必须设置 CLASSPATH 环境变量，或将 sourceanalyzer.jar 复制到 Ant 的 lib 目录下。

在 Ant 1.6 或更高版本中，也可以使用以下简单命令通过 Compiler Adapter 来运行 Ant：

```
sourceanalyzer -b <build-id> ant [ant-options]
```

默认情况下，系统会为 Fortify SCA 分配 600 MB 的内存用于转换。使用 Ant Compiler

Adapter 时，可以用-Dsourceanalyzer.maxHeap 选项来增加内存分配，如下所示：

```
ant -Dbuild.compiler=com.fortify.dev.ant.SCACompiler
-Dsourceanalyzer.buildid=MyBuild
-lib <install_directory>/Core/lib/sourceanalyzer.jar
-Dsourceanalyzer.maxHeap=1000M
```

### 4. 处理决议警告

在构建时为了看到所有生成的警告，在扫描之前输入以下命令：

```
sourceanalyzer -b <build_id> -show-build-warnings
```

### 5. Java 警告

可看到下面 Java 的警告：

Unable to resolve type...
Unable to resolve function...
Unable to resolve field...
Unable to locate import...
Unable to resolve symbol...
Multiple definitions found for function...
Multiple definitions found for class...

这些警告通常是由于缺少资源。例如，一些构建应用程序所需的 jar 和类文件没有指定。为解决警告，请确保包括应用程序将用到的所有必需文件已正确配置。

### 6. 使用 FindBugs

FindBugs (http://findbugs.sourceforge.net)是一个检测 Java 代码质量问题的静态分析工具。可将 Findbugs 与 Fortify SCA 一起使用，其结果会被合并到分析结果文件中。Findbugs 与 Fortify SCA 的不同之处在于，后者基于 Java 源文件运行，而前者基于 Java 字节码运行。因此，运行项目分析之前，应该首先编译项目并生成类文件。

为演示如何自动运行 FindBugs 与 Fortify SCA，编译代码示例 Warning.java，如下所示：

1) 切换到以下目录：

```
<install_directory>/Samples/advanced/findbugs
```

2) 输入以下命令并编译示例：

```
mkdir build
javac -d build Warning.java
```

3) 用 FindBugs 和 Fortify SCA 扫描该示例，如下所示：

```
sourceanalyzer -b findbugs_sample -java-build-dir build Warning.java
```

sourceanalyzer -b findbugs_sample -scan -findbugs -f
findbugs_sample.fpr

4) 在 Audit Workbench 中检查分析结果：

auditworkbench findbugs_sample.fpr

输出包括以下问题类别：

- Bad casts of Object References
- Dead local store
- Equal objects must have equal hashcodes
- Object model violation
- Unwritten field
- Useless self-assignment

如果按分析器分组，可以看到 Fortify SCA 结构分析器产生一条警告，而 FindBugs 产生 8 条。由 Fortify SCA 产生的 Object model violation 警告与由 FindBugs 产生的 Equalobjects must have equal hash codes 警告相似。此外，FindBugs 还会产生关于相同问题的两组警告(Useless self-assignment 和 Dead local store)。为避免结果重叠，需通过在扫描过程中使用-filter 选项来应用 filter.txt 过滤文件。注意，过滤并不十分精确，因为每个工具在不同的粒度级别上进行过滤。为演示如何避免结果重叠，使用 filter.txt 扫描示例代码，如下所示：

sourceanalyzer -b findbugs_sample -scan -findbugs -filter filter.txt
-f findbugs_sample.fpr

### 7. 转换 J2EE 应用程序

转换 J2EE 应用程序需要对 Java 源文件、J2EE 组件(如 JSP 文件)、部署描述符(如 web.xml)以及配置文件(如 struts-config.xml)进行分析。

具体步骤包括：

1) 转换 Java 文件。

请参考本章前面提到的示例(Java 命令行使用示例)。

2) 转换 JSP 文件。

3) 处理配置文件。

示例：

sourceanalyzer -b my_buildid "mydirectory/myfile.xml"

### 8. 转换 JSP 文件

要转换 JSP 文件，Fortify SCA 要求 JSP 文件采用标准的 Web Application Archive (WAR)布局。如果源目录已经采用 WAR 布局进行组织，那么可以直接从源目录中转换 JSP 文件。如果不是这种情况，那么需要部署应用程序并从部署目录中转换 JSP 文件。

如果 JSP 文件使用了任何标签库"例如 JSTL"请确保这些库的 jar 文件位于 WEB-INF/lib 目录中。否则 JSP 编译器将不处理标签库，并可能产生错误结果。

默认情况下，Fortify SCA 在转换阶段会使用某个版本的 Jasper JSP 编译器将 JSP 文件编

译为 Java 文件。然而，如果 Web 应用程序是专门针对某个应用程序服务器而开发的，那么在执行转换时必须使用该应用服务器的 JSP 编译器。

为了支持此操作，Fortify SCA 提供了以下命令行选项：

- -appserver 支持的值：weblogic/websphere
- -appserver-home

对于 Weblogic，包含 server/lib 子目录的目录路径

对于 WebSphere，包含 bin/JspBatchCompiler 脚本的目录路径

- -appserver-version 支持的值：

Weblogic 版本 7、8、9 和 10

WebSphere 版本 6

如果使用的应用程序服务器不属于这两种，请使用默认的内部 Fortify JSP 编译器。

例如：

在 Ant 1.6 或更高版本中，也可以使用以下简单命令通过 Compiler Adapter 来运行 Ant：

sourceanalyzer -b <build-id> ant [ant-options]

默认情况下，系统会为 Fortify SCA 分配 600MB 的内存用于转换。使用 Ant Compiler Adapter 时，可以用-Dsourceanalyzer.maxHeap 选项来增加内存分配，如下所示：

```
ant -Dbuild.compiler=com.fortify.dev.ant.SCACompiler
-Dsourceanalyzer.buildid=MyBuild
-lib <install_directory>/Core/lib/sourceanalyzer.jar
-Dsourceanalyzer.maxHeap=1000M
```

### 8.6.3　转换.NET 源代码

本节将描述如何使用 Fortify SCA 转换由以下程序构建的 Microsoft Visual Studio .NET 和 ASP.NET 应用程序：

- NET 版本 1.1 和 2.0
- Visual Studio .NET 版本 2003
- Visual Studio .NET 版本 2005

Fortify SCA 基于 Microsoft 中间语言(MSIL)运行，因此支持所有编译成 MSIL 的.NET 语言，包括 C# 和 VB .NET。

包括以下主题:

- Visual Studio .NET
- 转换简单的.NET 应用程序
- 转换 ASP.NET 1.1 (Visual Studio 版本 2003)项目

注意：分析.NET 应用程序最简单的方法是，使用适用于 Visual Studio 的 Fortify Secure CodingPlug-in，可自动收集有关项目的信息。

#### 1. Visual Studio .NET

如果使用 Visual Studio .NET 执行构建命令行，可通过调用 sourceanalyzer 来封装构建命令行，从而轻松地集成源代码分析。为此，具有的安全编码规则包必须适用于当前所安装的

VisualStudio 版本。

以下示例显示了适用于 Visual Studio .NET 的命令行语法：

sourceanalyzer -b my_buildid Sample1.sln /REBUILD debug

它在 Visual Studio 构建的所有文件中执行转换阶段。务必进行清除或重新构建，确保将所有文件都包含在内。然后就可以执行分析阶段了，如下所示：

sourceanalyzer -b my_buildid -scan -f results.fpr

### 2. 转换简单的.NET 应用程序

用户可使用 Fortify SCA 命令行界面来处理.NET 应用程序。

使用以下方法之一为应用程序的分析工作做准备：

- 在启用“调试”配置的情况下执行完整的项目重建。在启用调试的情况下编译项目可提供 Fortify SCA 显示结果所需的信息。
- 获取与项目有关的所有第三方.dll 文件、项目输出.dll 文件和相应的.pbd 文件。注意，如果同一文件夹中不存在相应的.pdb 文件，Fortify SCA 就会忽略任何作为输入变量传送的.dll 文件。因此，必须包含与所有项目.dll 文件对应的所有.pdb 文件。

注意：.pdb 文件在第三方库中并不是必要的。

从命令行运行 Fortify SCA 来分析.NET 应用程序，如下所示：

- 对于 Visual Studio .NET 版本 2003，输入：

sourceanalyzer -vsversion 7.1 -b MyBuild

-libdirs ProjOne/Lib;ProjTwo/Lib ProjOne/bin/Debug ProjTwo/bin/Debug

其中：

MyBuild 是构建标识符；

ProjOne/Lib;ProjTwo/Lib 是用分号分隔的文件或 DLL (包含第三方 DLL)路径列表；

ProjOne/bin/Debug ProjTwo/bin/Debug 是输出文件夹；

对于 Visual Studio .NET 版本 2005，输入：

sourceanalyzer -vsversion 8.0 -b MyBuild

-libdirs ProjOne/Lib;ProjTwo/Lib ProjOne/bin/Debug ProjTwo/bin/Debug

其中：

MyBuild 是构建标识符；

ProjOne/Lib;ProjTwo/Lib 是用分号分隔的文件或 DLL (包含第三方 DLL)路径列表；

ProjOne/bin/Debug ProjTwo/bin/Debug 是输出文件夹；

注意：Fortify SCA 会自动获取项目中所用的标准.NET DLL 文件，因此命令行中不必包含这些文件。

如果项目很大，可以使用相同的构建 ID 对每个输出文件夹分别执行转换阶段，如下所示：

sourceanalyzer -vsversion <version_number> -b <build_id>

-libdirs <paths><folder_1>

...

sourceanalyzer -vsversion <version_number> -b <build_id>

-libdirs <paths><folder_n>

其中：

<version_number>既可以是 7.1 也可以是 8.0；

<build_id>是构建 ID；

<paths>是带有第三方 DLL 文件的文件夹或 DLL 文件路径的列表，列表中的各项内容以分号分隔，<folder_1>和<folder_n>是输出文件夹；

注意：Fortify SCA 需要配合正确的 Visual Studio 版本，即使使用的是命令行界面。

### 3. 转换 ASP.NET 1.1 (Visual Studio 版本 2003)项目

如上所述，Fortify SCA 基于.NET 编译器所生成的 MSIL 运行。对于 ASP.NET 项目，Web 组件(如.aspx 文件)需要先编译然后再分析。然而，目前没有针对.aspx 文档的标准编译器，当从浏览器中打开它们时，.NET 1.1 运行时会自动对其进行编译。

为辅助.aspx 编译阶段，Fortify Software 提供了一个简易工具，用于编译项目中的所有.aspx 文件。该工具在 Fortify 安装目录中的位置是：

\Tools\fortify_aspnet_compiler\fortify_aspnet_compiler.exe

分析 ASP.NET 1.1 解决方案：

1) 对解决方案执行完整的重新构建。

2) 对于解决方案中的每个 Web 项目，删除以下文件夹：

%SYSTEMROOT%\Microsoft.NET\Framework\v1.1.4322\Temporary ASP.NETFiles\<web_application_name>

3) 对于解决方案中的每个Web项目，运行以下命令：

fortify_aspnet_compiler <url_to_the_web_site>

<source_root_of_the_web_project>

其中：

<url_to_the_web_site>指网站的 URL，例如：

http://localhost/WebApp

<source_root_of_the_web_project>指 Web 项目的源位置，例如：

<VS_project_location>\WebApp

4) 为在步骤 1 中构建的 DLL 执行转换阶段。键入以下命令，其中使用的构建 ID 与以下步骤中的相同：

sourceanalyzer -b <build_id>"<VS_project_location>***.dll"

5) 对 Web 组件执行转换阶段。对于解决方案中的每个 Web 项目，请在调用 sourceanalyzer 时输入以下命令：

sourceanalyzer -b <build_id>

%SYSTEMROOT%\Microsoft.NET\Framework\v1.1.4322\Temporary

ASP.NETFiles\<web_application_name>

6) 包含配置文件和现有的任何 Microsoft T-SQL 源文件：

sourceanalyzer -b <build_id>"<solution_root>***.config"

<"t-sql_src>***.sql">

注意：如果使用的是 Microsoft Visual Studio 安全编码包，那么这些步骤都将自动执行。

### 8.6.4　转换 C 和 C++代码

本章将介绍如何使用 Fortify SCA 来转换 C 和 C++源代码以进行分析。

#### 1. C 和 C++ 命令行语法

转换一个文件所用的基本命令行语法是：

sourceanalyzer -b <build_id><compiler> [<compiler options>]

其中：

- <compiler>是在项目的构建扫描过程中使用到的编译器名称，如 gcc 或 cl。
- <compiler options>是传送到编译器通常用于编译文件的选项。

#### 2. C 和 C++ 命令行示例

以下是一个简单用法示例：

要使用gcc 编译器转换一个名为helloworld.c 的文件，请输入：

sourceanalyzer -b my_buildid gcc helloworld.c

注意：这个过程中会编译文件。

#### 3. 与 Make 集成

可以使用以下任一方法配合使用 Make 和 Fortify SCA：

- 使用 Fortify Touchless Build Adapter
- 修改 Makefile 以调用 Fortify SCA

#### 4. 使用 Fortify Touchless Build Adapter

使用 Fortify Touchless Build Adapter 来集成 makefiles，请运行以下命令：

sourceanalyzer -b <build_id> touchless make

Fortify SCA 会运行 make 命令。当 make 调用了任意被 Fortify SCA 认为是编译器的命令时，Fortify SCA 就会处理该命令。注意，makefile 不会被修改。

这种构建集成的方法并不局限于 make。任何执行编译器处理的构建命令都适用于该系统，只要将用于运行构建的命令替代上述命令中的'make' 部分即可。

注意：如果存在以下情况，Fortify touchless build adapter 会表现异常：

- 构建脚本采用绝对路径调用编译器，或者构建脚本重写了可执行的搜索路径；
- 构建脚本未创建新的进程来运行编译器。许多 Java 构建工具(包括 Ant)都以这种方式运行。

#### 5. 修改 Makefile 以调用 Fortify SCA

要修改 makefile 以调用 Fortify SCA，请将 makefile 中对编译器、存档文件或链接的调用替换为对 Fortify SCA 的调用。这些工具通常在 makefile 的一个特殊变量中指定，如下所示：

```
CC=gcc
CXX=g++
AR=ar
```

该步骤可简化为在 makefile 中与 Fortify SCA 一起预先计划对这些工具的引用以及相应的选项：

CC=sourceanalyzer -b mybuild -c gcc

CXX=sourceanalyzer -b mybuild -c g++

AR=sourceanalyzer -b mybuild -c ar

### 6. 使用 Fortify Build Monitor

本节将介绍在 Windows 系统上进行构建的过程中，如何使用 Fortify Build Monitor 自动扫描 C/C++项目并查看结果。其中示例使用的是 Fortify SCA 随附的项目范例。

Fortify Build Monitor 概述

Fortify Build Monitor 菜单包含表 8-2 所示的选项。

表 8-2　Fortify Build Monitor 的选项

| 选　　项 | 描　　述 |
| --- | --- |
| Monitor (监视器) | 启动监视。Build Monitor 会拦截并转换计算机上的下一个构建 |
| Build Done (构建完成) | 在构建完成后停止监视 |
| Scan (扫描) | 扫描在构建过程中监视的代码 |
| Scan Settings (扫描设置) | 控制规则包和内存设置 |
| Set Results Folder (设置结果文件夹) | 控制 Fortify SCA 输出结果的位置 |
| Stay on Top (始终在上面) | 使 Fortify Build Monitor 窗口始终位于其他窗口之上 |
| Minimize to Tray (最小化到任务栏中) | 将 Fortify Build Monitor 显示为任务栏中的一个图标 |
| Exit (退出) | 关闭 Fortify Build Monitor |
| Show Messages (显示消息) | 在窗口下方区域显示或隐藏消息。具体的消息包括扫描消息、错误消息和 Monitor Driver 信息。可以单击窗口底部的 Detailed Messages (详细消息) |
| Help (帮助) | 显示联机帮助 |
| Reset (重置) | 将 Fortify Build Monitor 重置为初始状态 |

#### 配置 Fortify Build Monitor

1) 设置结果文件夹

Fortify Build Monitor 将结果以 FPR 格式输出到一个本地文件夹中。可以更改输出文件夹。FortifyBuild Monitor 在每次执行扫描后会覆盖结果。这些结果不会存档。

更改结果文件夹的步骤：

a) 选择 Action (操作)，再选择 Set Results Folder (设置结果文件夹)。

屏幕上将显示 Browse for Folder (浏览文件夹)对话框。

b) 选择一个文件夹，然后单击 OK (确定)。

Fortify Build Monitor 将结果输出到选定的文件夹中。

2) 设置 Fortify SCA 扫描选项

Fortify Build Monitor 使用 Fortify SCA 扫描项目。可以调整以下扫描设置：

- Allocate memory (分配内存)：增加或减少分配给 Fortify SCA 的内存；
- Secure coding rulepacks and custom rulepacks (安全编码规则包和自定义规则包)：更改 FortifySCA 用来分析源代码的规则包；
- User (用户)：仅监视由当前用户执行的构建。

更改扫描选项的步骤：

a) 选择 Action(操作)，再选择 Scan Settings (扫描设置)。

屏幕上将显示Fortify Build Monitor: Scan Settings (Fortify Build Monitor：扫描设置)对话框。

b) 要更改内存分配，请选择一个值。

注意：如果输入的选项无效，则会把内存设置为无限大。

c) 要添加或删除规则包，请单击 Rulepacks (规则包)。

d) 要查看 Fortify SCA 命令行选项，请单击 Preview (预览)。

e) 单击 Done (完成)。

Fortify SCA 扫描选项完成更改。

监视构建

对于 Windows 平台上的 C/C++项目和解决方案，Fortify SCA 包含 Fortify Build Monitor，该工具包含图形化用户界面，可在构建过程中自动执行分析。

在 Windows 平台上分析 C/C++ 源代码构建的步骤：

a) 选择“开始”，然后依次选择“所有程序”和 Fortify Software-Fortify SCA - Build Monitor。

b) 单击 Monitor (监视器)。

当监视器启动后，会显示一个绿色指示灯图标。

c) 在构建环境中创建完整的项目构建。

d) 检查是否已成功完成构建。

e) 返回 Fortify Build Monitor 窗口，然后单击 Build Done (构建完成)。

f) Fortify SCA 会将结果输出到一个子文件夹中，请为该输出文件夹指定名称。如果该文件夹已经存在，则 Fortify SCA 会在开始扫描之前清理该文件夹。

g) 单击 Scan (扫描)。

Fortify SCA 会显示结果并将其存储在指定文件夹下的一个 FPR 文件中。

注意，要查看结果，可在 Audit Workbench 平台或者使用 Microsoft Visual Studio 安全编码包打开该 FPR 文件。

监视项目的示例

此示例是 Windows 用户对名为 qwik-smtpd 的 C++ 代码项目示例进行分析。此示例使用了 Microsoft Visual Studio 和 Fortify Build Monitor。

分析 qwik-smtpd 项目：

a) 使用 Microsoft Visual Studio，打开并构建 Tutorials/C/source 目录下的 qwik-smtpd 项目。

b) 选择“开始”，然后依次选择“所有程序”和 Fortify Software - Fortify SCA - Build Monitor。

c) 单击 Monitor (监视器)。

d) 将该窗口最小化。

e) 在 Microsoft Visual Studio 中，重新构建该项目。

注意, 由于项目中没有发生任何变化，因此必须使用重新构建选项。

f) 检查是否已成功完成构建。

g) 返回 Fortify Build Monitor 窗口，然后单击 Build Done (构建完成)。

h) 指定构建输出的位置。

i) 单击 Scan (扫描)。

Fortify SCA 会在指定的文件夹下保存一个FPR 文件。

注意，要查看结果，请在 Audit Workbench 平台或者使用 Microsoft Visual Studio 安全编码包打开该 FPR 文件。

### 7. Visual Studio .NET

如果使用 Visual Studio .NET 来执行命令行构建，只要通过封装构建命令行并调用 sourceanalyzer，即可轻松集成 source code analysis。为此，必须在 Visual Studio 版本中安装一个 Fortify 安全编码插件。

观察以下示例：

sourceanalyzer -b my_buildid devenv /REBUILD MyProject.sln

它在 Visual Studio 构建的所有文件中执行转换阶段。务必进行清除或重新构建，确保将所有文件都包含在内。

### 8. Visual Studio 6.0

如果用 Visual Studio 6.0 来执行命令行构建，可通过封装构建命令行并调用 sourceanalyzer 来集成 source code analysis。

观察下面这个例子：

sourceanalyzer -b my_buildid msdev MyProject.dsp /MAKE "MyProject DEBUG"/REBUILD

这会对 Visual Studio 构建的所有文档都执行转换阶段。正如 Visual Studio 文档中所描述的，为将所有文件都包含在内，务必要进行清除或重新构建。

## 8.6.5　转换 ABAP/4

### 1. 关于转换 ABAP/4 代码

翻译 ABAP/4 代码类似于翻译其他操作语言的代码，但需要额外的步骤从 SAP 数据库中提取代码，并准备将其扫描。本章假设 SCA 正在运行，并且读者对 SCA、SAP 和 ABAP/4 有一个基本了解。

### 2. 扫描 ABAP 代码

当一个包从 ABAP 被提取出来，HP Fortify 的 ABAP 提取器用与包名匹配的 parentcl 域从 TDEVC 中提取所有数据，然后把提取出的数据作为 parentcl 域，同之前一样再递归从 TDEVC 提取相关所有数据。从 TDEVC 提取的这个字段称为 devclass。DEVCLASS 值被视为一组程序名，并与同其他程序名一样执行相关的操作。

程序是通过匹配名称域从 TRDIR 提取得到，该名称域可能是用户在选择屏幕中提供的程序名得到，也可能通过与来自 TDEVC 提取的值的列表作对比得到(如果提供了一套给定的匹配名称字段中的程序名称)。从 TRDIR 获得的行就是那些名称域，它们具有像 programname 的程序名称，名称域的作用是继续提取行。最终名单是 READ　REPORT 用于从 SAP 系统获取代码。此方法仅从 REPORTs 读取类和方法。

每个 READ　REPORT 调用产生一个文件，此文件在本地系统上的临时文件夹中。这组文件由源分析器翻译和扫描，产生可被 HP Fortify 的 Audit Workbench 查看的 FDR 文件。

当源代码下载完成，HP Fortify 的 ABAP 提取器会检查 INCLUDE 描述语句。被发现时，它也会将 INCLUDE 内容下载到本地计算机进行分析。

### 3. 过程概述

在翻译 ABAP/4 代码之前需要两个主要步骤：

- SAP 服务器上安装一个传输请求，在这个步骤中，将安装 HP Fortify 的提取程序；
- 创建一个事务对象。

### 4. 传输请求

ABAP 扫描可作为 SCA 的高级功能，如果购买了包含此功能的许可证，需要在 SAP 服务器上安装 HP Fortify 的传输请求，请联系 HP Fortify 技术支持部门获取一份惠普传输请求或相关信息的副本，在 SCA 的许可副本中再添加这个功能。

### 5. 创建事务对象

将需要为推出一个 SCA 扫描创建一个事务对象。按下面的步骤来创建一个事务对象。

1) 按 Esc 键，直至到达 SAP 轻松访问屏幕，并键入 se93。Maintain Transaction 界面出现，如图 8-24 所示。

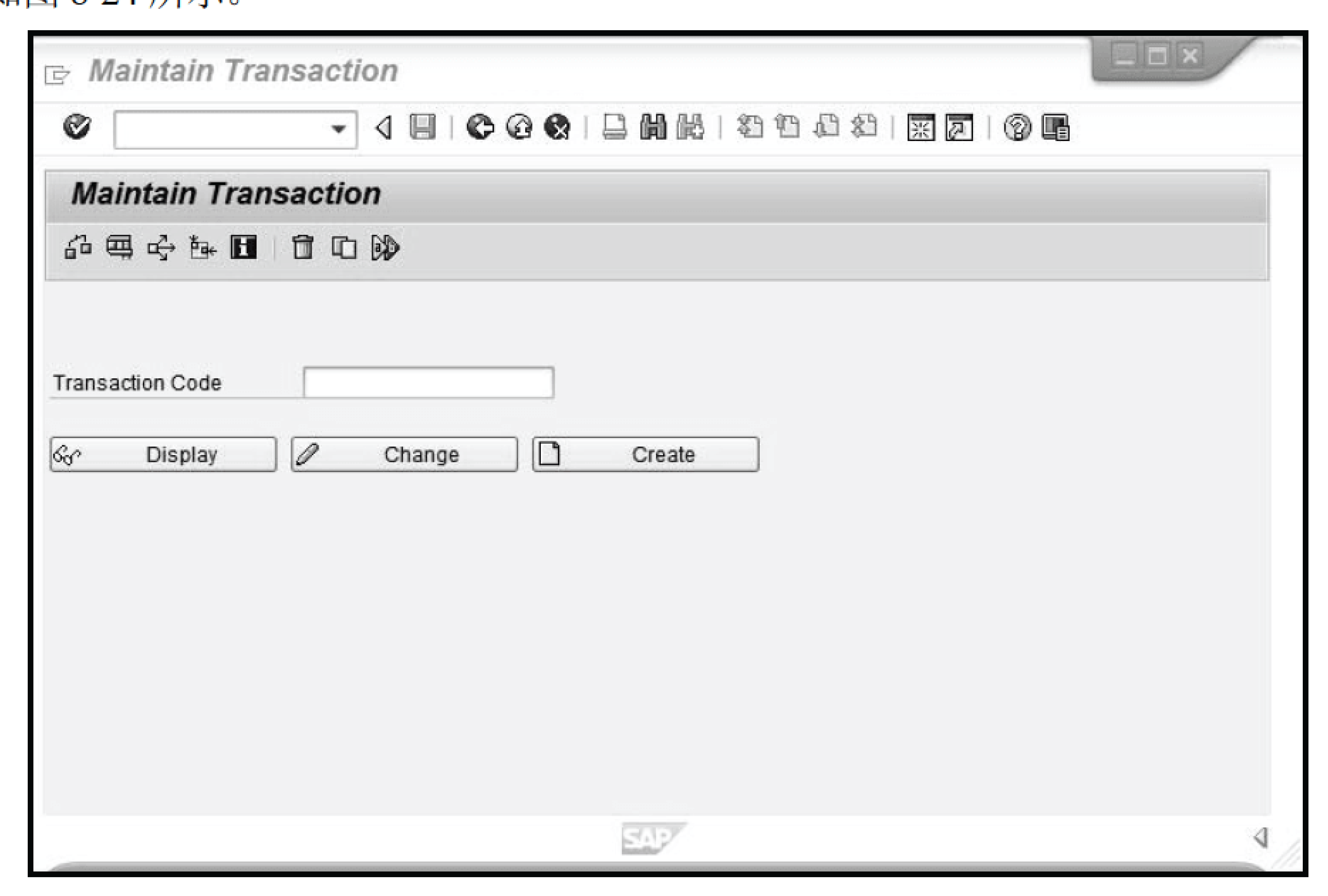



图 8-24　Maintain Transaction 界面

2) 在事务代码框中键入 Z_AMOL_SCA 并单击 Create 按钮。将出现 Create Transaction 对话框，如图 8-25 所示。

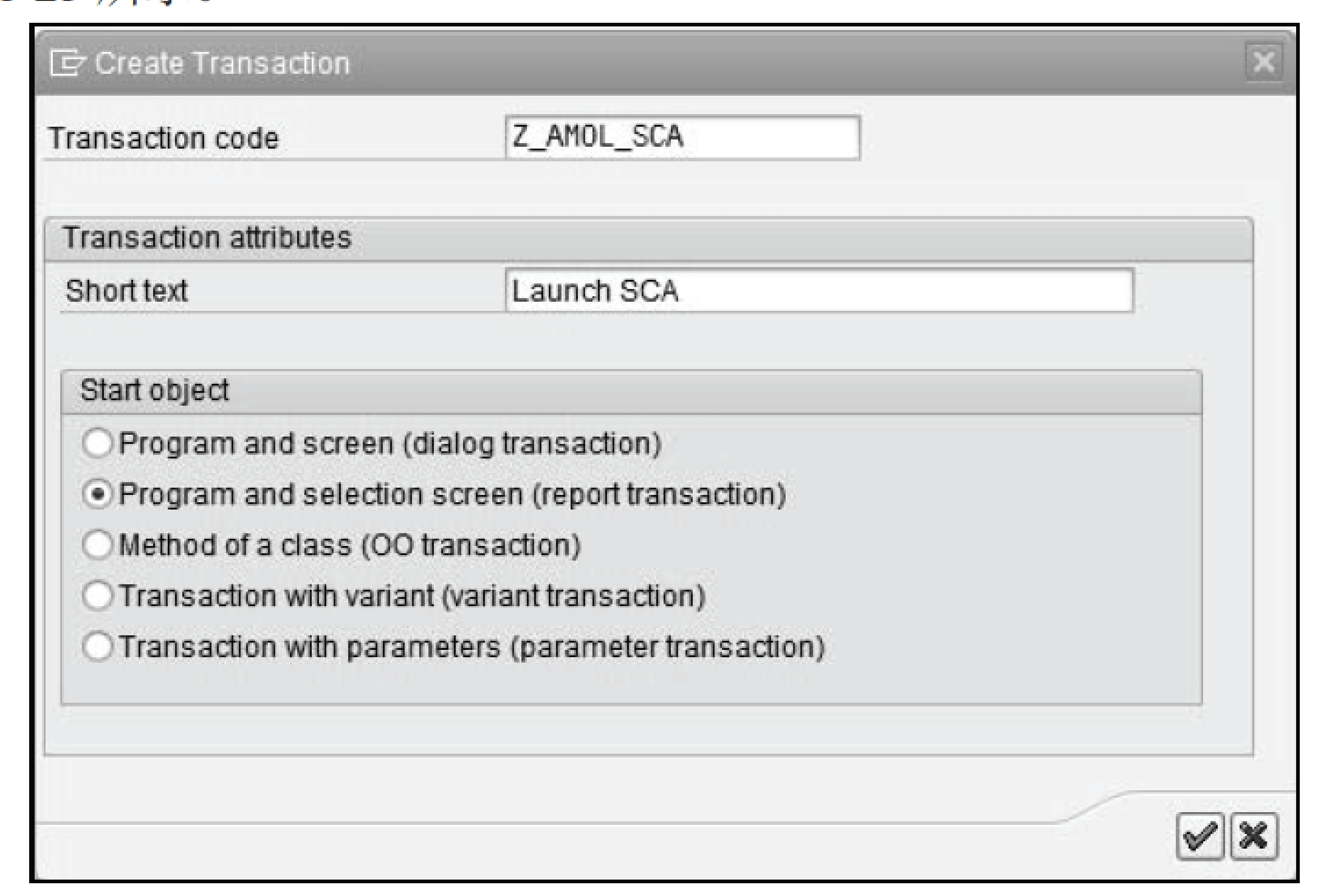



图 8-25　Create Transaction 界面

3) 在 Transaction attributes 部分，在短文本框中输入事务(例如：运行 SCA)目标的简短说明。

4) 在 Start object 部分，选择程序和选择屏幕(报告交易)单选按钮。

5) 单击绿色的对号按钮，出现 Create Report Transaction 屏幕，如图 8-26 所示。

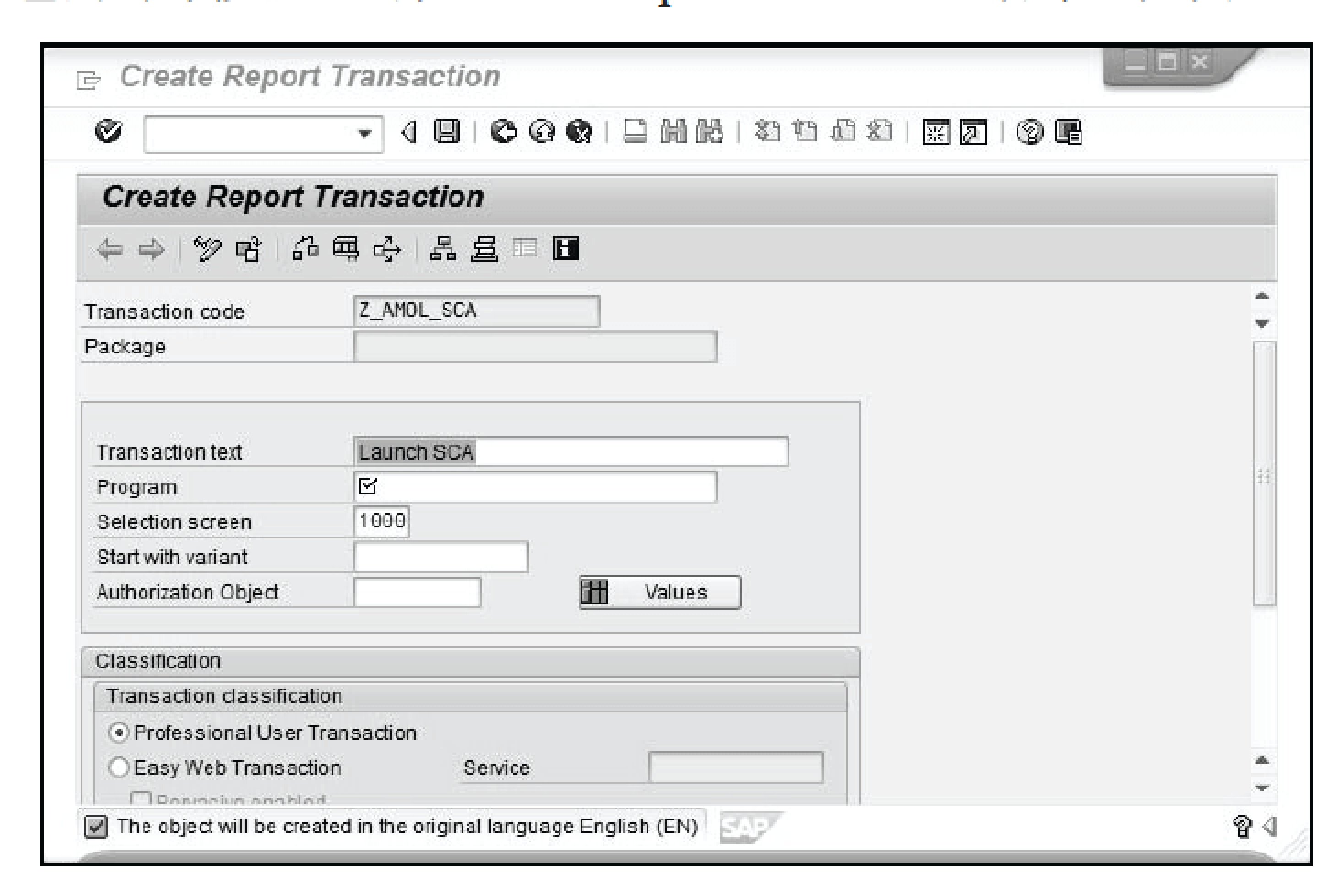



图 8-26　Create Report Transaction 界面

6) 在 Package 文本框中键入$TMP。

注意，如果 Package 框不可编辑，请空着，在之后的界面可输入 Package 详细信息。

7) 在 Program 框输入程序名称(例如 Z_AMOL_SCA)

8) 在图像支持部分，选择所有三个复选框。

9) 单击工具栏上的 Save。Create Object Directory Entry 界面会显示出来，如图 8-27 所示。

注意：如果未在 Package 字段中输入一个屏幕中包的详细信息，请键入$ TMP。

10) 击 Save 按钮。

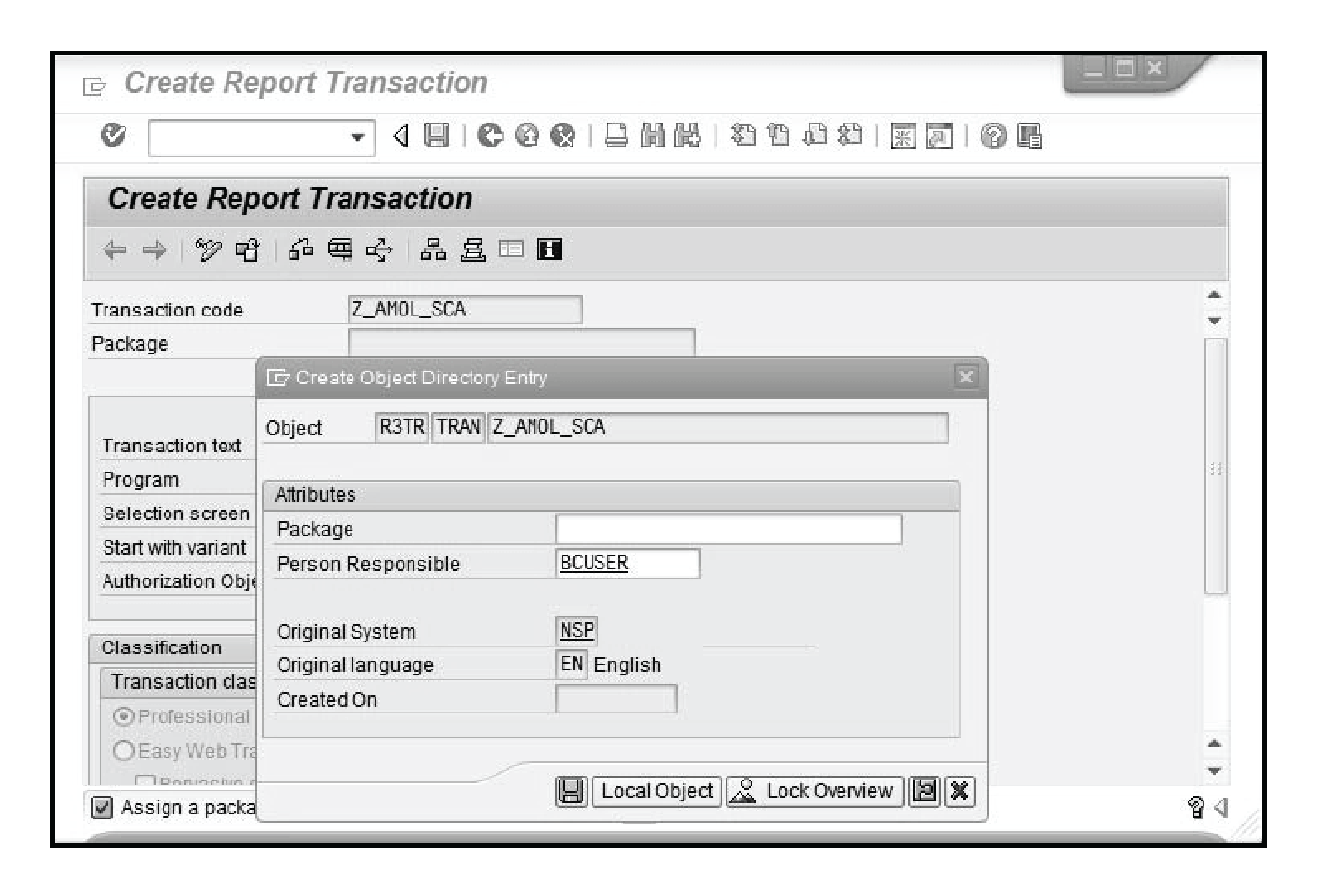



图 8-27　CreateObject Directory Entry 界面

## 6. 将 Fortify SCA 加入收藏列表

将 Fortify SCA 添加到收藏夹列表是可选项，但这样做可能会更便捷地进入和启动 Fortify SCA 扫描。下面的步骤假定是日常使用的用户菜单。如果是从不同菜单进行，将收藏夹链接添加到使用的菜单。再创建的 Fortify SCA 项目，SAP 服务器将在基于 Web 的客户端的 SAP 轻松访问区域运行。

1) 从 SAP 轻松访问菜单，在会话框键入 S000，SAP 菜单出现。

2) 右击最爱文件夹，选择插入会话。Manual entry of a transaction 框出现，如图 8-28 所示。

3) 在会话代码框键入 Z_AMOL_SCA。

注意：如果创建交易代码时选择了不同的名称，请使用这个名称。

4) 单击绿色对号按钮，运行 SCA 选项出现在收藏列表中，如图 8-29 所示。

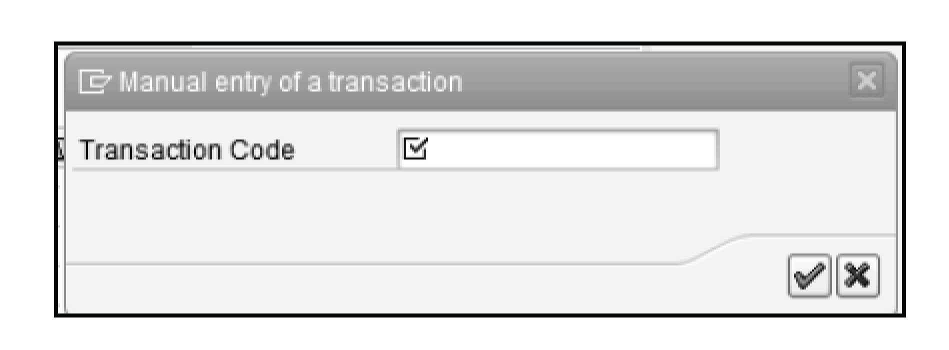



图 8-28　Manual entry of a transaction 界面

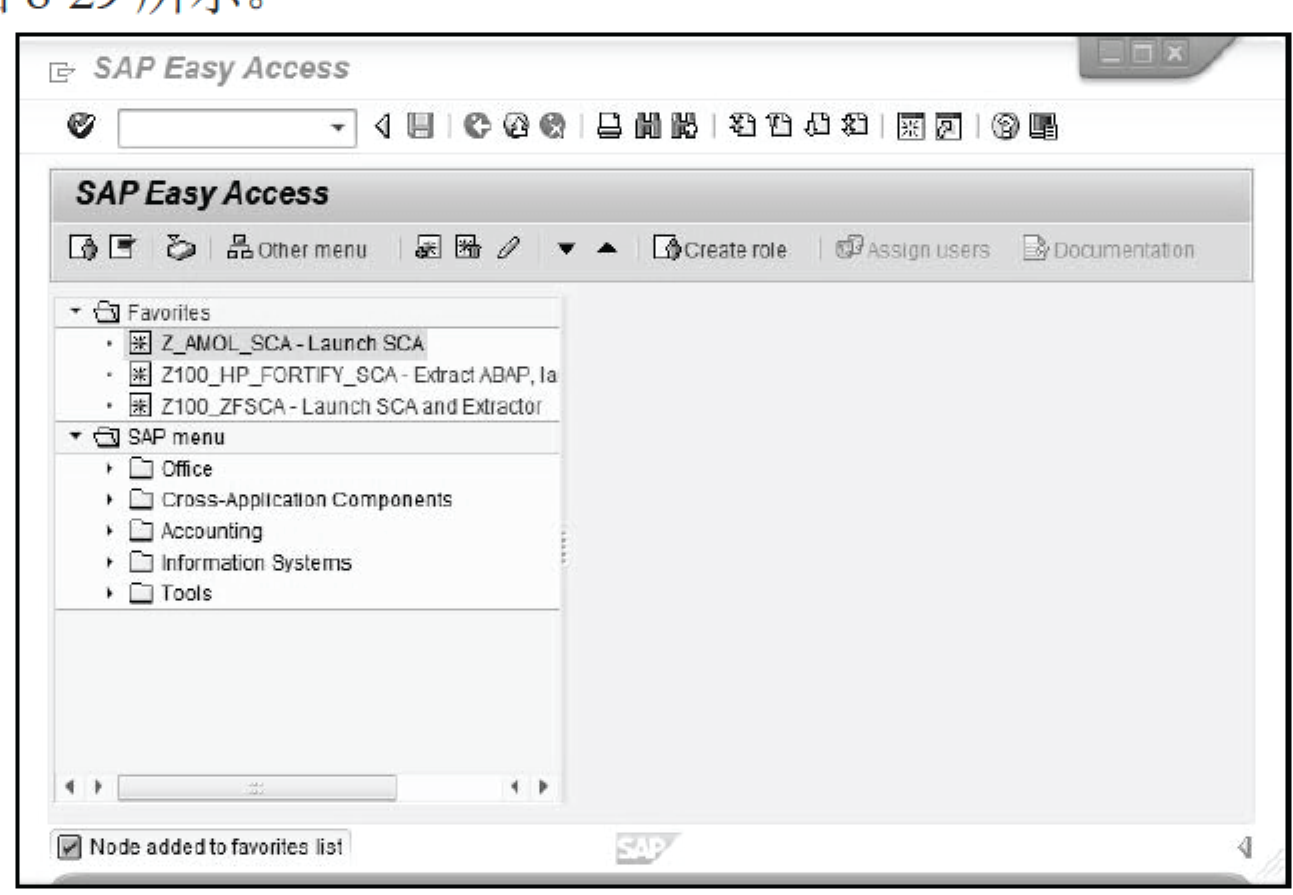



图 8-29　收藏列表界面

单击 link 启动 SCA，如图 8-30 所示。

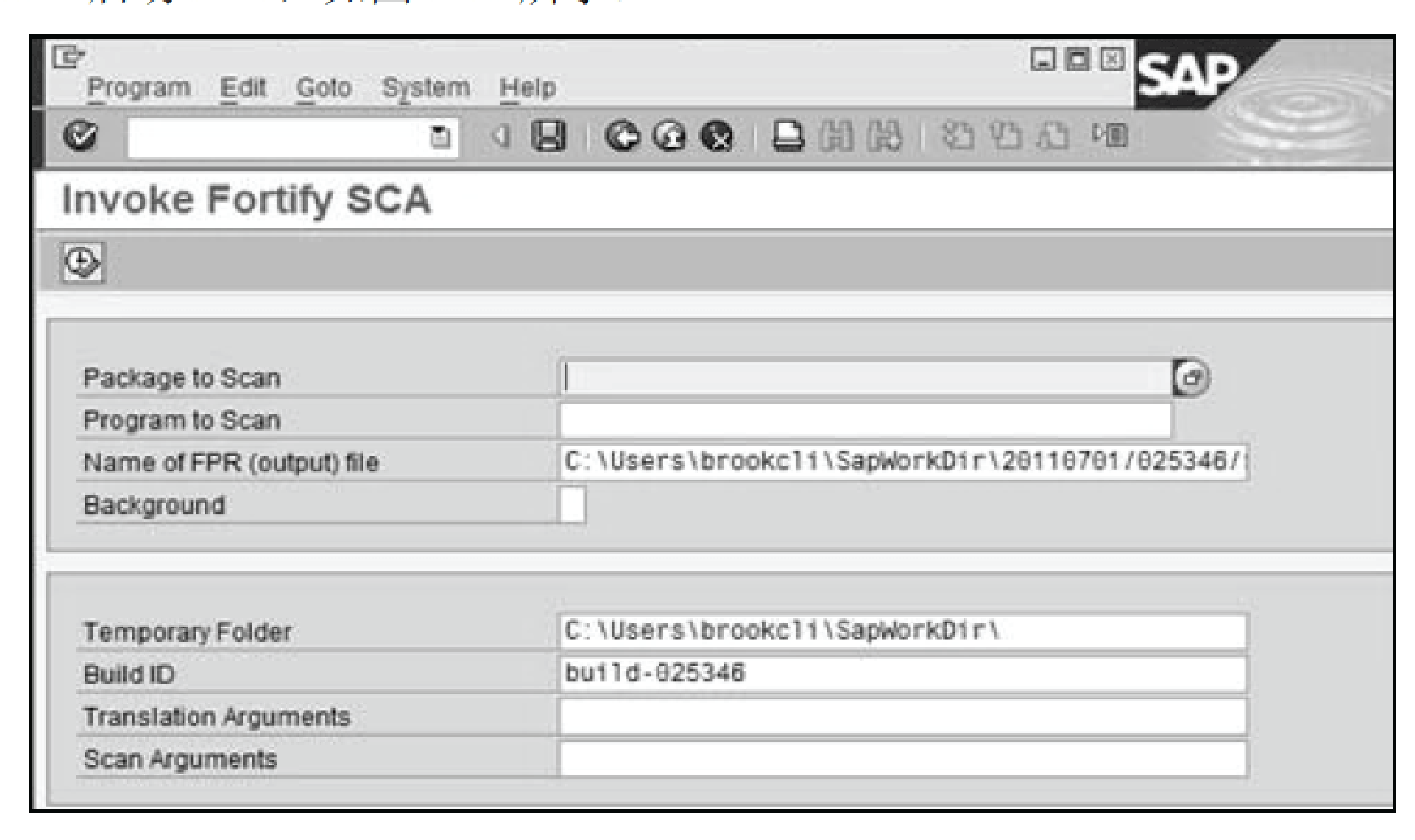



图 8-30　Invoke Fortify SCA

## 7. 运行 HP Fortify ABAP 提取器

1) 从 Favorites 链接运行程序(交易代码)，或手动启动 Z_AMOL_SCA 对象，如图 8-31 所示。

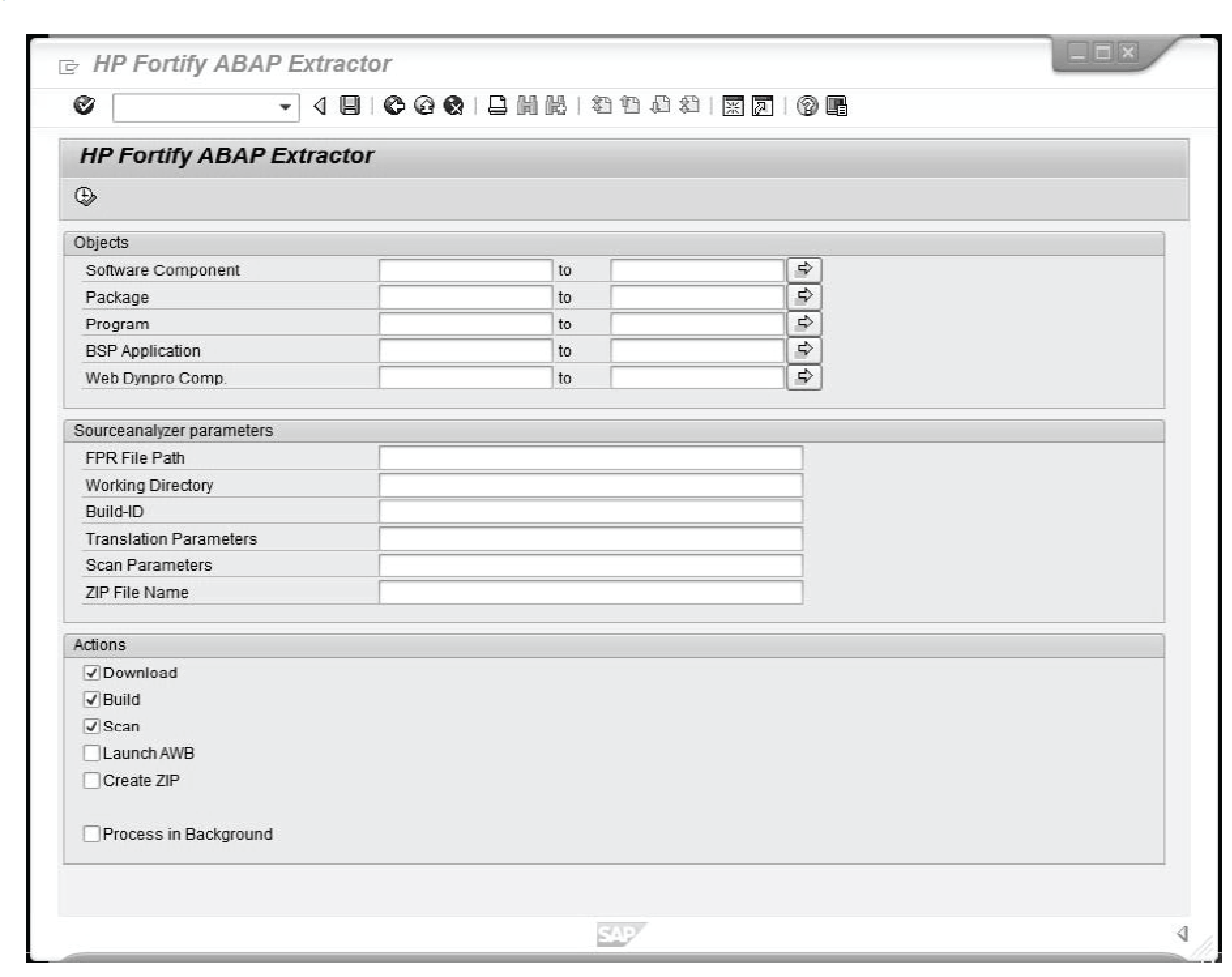



图 8-31　HP Fortify ABAP 提取器

2) 填写所要求的信息，如表 8-3 所示。

表 8-3　要求填写的信息

| 部　分 | 数　据 |
| --- | --- |
| 对象 | 输入想要扫描的软件组件、包、程序、BSP 或 Web Dynpro 组件名 |
| 源分析器参数 | FPR 文件路径：输入想要存储的 FPR 文件路径，包括要指定的 FPR 文件名称。<br>工作目录：输入提取的源代码所在的目录并复制此目录。<br>构建 ID：输入构建 ID 为扫描。<br>转译参数：列出所有可选源分析器转换参数。<br>扫描参数：列出所有可选源分析器扫描参数。<br>ZIP 文件名：如果需要压缩包提供输出，输入一个 ZIP 文件名 |
| 动作 | 下载：选中此复选框以指示 SCA 下载从 SAP 数据库中提取的源代码。<br>建设：选中此复选框用<>符号以指示 SCA<br>扫描：勾选此项，要求扫描。<br>启动 AWB：选中此复选框来启动审计工作台并加载 FPR。<br>创建 ZIP：选中此复选框，要求输出是压缩格式。<br>过程中背景：选中此复选框以处理发生在后台的请求 |

3) 单击执行按钮。

## 8.6.6　转换 FLEX

### 1. 命令

下面列出命令行选项：

-flex-sdk-root(com.fortify.sca.FlexSdkRoot)应指向一个有效的 Flex SDK 的根。此文件夹应包含 flex-config.xml 的一个框架文件夹。它也应该包含一个 bin 文件夹(包含一个 mxmlc 可执行文件)。用户可在-sca.properties 文件中设置此属性。

-flex-libraries(com.fortify.sca.FlexLibraries)包含“：”或“；”分隔的列表(“：”为大多数平台，“；”为 Windows 平台)上要“链接”到库的名称。大多数情况下，该列表包括 flex.swc，framework.swc 和 playerglobal.swc(通常在 Flex SDK 根目录下的 frameworks/libs/)。用户可在-sca.properties 文件中设置此属性来使用 SWC 的相同设置。

注意：用户可指定 SWC 或 SWF 文件作为 Flex 库，但目前不支持 SWZ。

-flex-source-roots(com.fortify.sca.FlexSourceRoots)包含“：”或“；”在 MXML 源都可以找到的根目录中分隔的列表。通常情况下，这些都将包含一个名为 com 的子文件夹。例如，如果一个 Flex 源指向 foo/bar/src 目录，foo/bar/src/com/fortify/manager/util/Foo.mxml 将得到一个名为 com.fortify.manager.util.Foo 的对象(在 com.fortify.manager.util 包中的名为 Foo 的对象)。

-flex-sdk-root 和-flex-source-roots 主要用于 MXML 编译，并且为可选，如果你扫描纯 ActionScript。-flex-libraries 用于解决所有的 ActionScript。

注意，MXML 文件被翻译成 ActionScript，然后通过动作脚本解析器运行。所生成的 Action Script 旨在进行简单分析，不会严格纠正，如 Flex 的运行时模型。由于这一结果，可能会得到 MXML 文件解析错误。例如，XML 解析可能会失败，翻译为 ActionScript 可能会失败，

以及由此产生的 ActionScript 的解析也可能会失败。如果发现任意与源代码无关的错误，请通知 HP Fortify 的技术支持部门。

### 2. ActionScript 命令行语法

执行 ActionScript 的基本命令行语法是：

sourceanalyzer -b <build_id> -flex-libraries <listOfLibraries>

要将文件直接传递到 Fortify SCA，请输入：

sourceanalyzer -b <build_id> -flex-libraries <listOfLibraries>

### 3. ActionScript 命令行举例

例子 1

下面的例子是只包含一个 MXML 文件和单个 SWF 库(MyLib.swf)的简单应用。

sourceanalyzer -b MyFlexApp -flex-libraries lib/MyLib.swf -flex-sdk-root /home/myself/flex-sdk/ -flex-source-roots.my/app/FlexApp.mxml

以上标识库的位置，包括并确定了 Flex SDK 和 Flex 源的根目录位置。这个位于/my/app/ FlexApp.mxml 的独立 MXML 文件被转换成位于 my.app 包下名为 FlexApp 的一个独立 ActionScript 类。

例子 2

下面的例子是在源文件是相对于 src 目录的应用程序。它使用一个单一的 SWF 库 MyLib.swf，以及 Flex 和 Flex SDK 的框架库。

sourceanalyzer -b MyFlexProject -flex-sdk-root /home/myself/flex-sdk/ -flex-sourceroots src/ -flex-libraries lib/MyLib.swf src/**/*.mxml src/**/*.as

在这个例子中，实现定位 Flex SDK。SCA 文件说明符用来包含 src 文件夹下的.as 和 mxml 文件。这指定在-flex-sdk-root 下没必要显式指出的.swf 文件。但本例做法是为了进行说明。SCA 会自动定位所有Flex SDK根目录下.swc文件，并假定这些都旨在用于翻译的ActionScript 或 mxml 文件。

例子 3

在这个例子中，Flex SDK 的根和 Flex 库在属性中被指定，因为输入数据文件非常耗时，该应用程序可分成并保存为两部分：一个主要部分文件夹和一个模块文件夹。每个文件夹应包含其开始路径的src 文件夹。使用通配符的文件规范获取所有.mxml 和.as 文件。main/src/com/foo/util/Foo.mxml 将被转换为包 com.foo.util 中名为 Foo 的一个 ActionScript 类，例如，用在此处指定的源根目录：

sourceanalyzer -b MyFlexProject -flex-source-roots main/src:modules/src ./main/src/**/*.mxml ./main/src/**/*.as ./modules/src/**/*.mxml ./modules/src/**/*.as

### 4. 处理警告

要查看构建过程中生成的所有警告，启动扫描之前请输入以下命令：

sourceanalyzer -b <build_id> -show-build-warnings

可能会收到类似的消息：

The ActionScript front end was unable to resolve the following imports: a.b at y.as:2. foo.bar at somewhere.as:5. a.b at foo.mxml:8.

当 SCA 无法找到所有需要的库，会发生此错误。可能需要指定额外的 swf 与 swfFlex 库 (-FLEX-libraries 选项，或 com.fortify.sca.Flex 库属性)，使 SCA 可以完成分析。

## 8.6.7　转换移动平台代码

### 1. 转换 Objective-C 代码

1) Objective-C 命令行语法

用于转换单个文件中的基本命令行语法

sourceanalyzer -b <build_id><compiler> [<compiler options>] -clean

sourceanalyzer -b <build_id><compiler> [<compiler options>]

- <compiler>是生成项目的扫描过程中使用的编译器名。要扫描完整的 Xcode 项目，可以用 xcodebuild 作为编译器。扫描单独的源文件，可通过 clang 或 llvm-gcc 作为编译器。
- <compiler options>是通常用于编译文件的编译器选项。

2) Objective-C 命令行例子

下面的简单例子说明了所支持的编译器的使用模式。下面的命令应当在项目文件所在的目录下运行。

使用 Xcode 编译器转换名为 myproject 的工程，请输入：

sourceanalyzer -b my_buildid xcodebuild -sdk iphonesimulator -clean

sourceanalyzer -b my_buildid xcodebuild -sdk iphonesimulator

如果你有苹果开发者证书，传递-sdk iphoneos 而不是-sdk iphonesimulator。

使用 gcc 编译器转换名为 HelloWorld.m 文件，请输入：

sourceanalyzer -b my_buildid llvm-gcc -x objective-c HelloWorld.m -clean

sourceanalyzer -b my_buildid llvm-gcc -x objective-c HelloWorld.m

使用 clang 编译器转换名为 HelloWorld.m 文件，请输入：

sourceanalyzer -b my_buildid clang -ObjC HelloWorld.m -clean

sourceanalyzer -b my_buildid clang -ObjC HelloWorld.m

扫描应用程序文件

sourceanalyzer -b my_buildid -scan -f result.fpr

注意，运行这些命令时，源代码将被编译。

Xcode 编译器错误

如果收到 Xcode 编译器错误，这可能是由于 SCA 新版本新增 clang 选择的原因。为消除这些错误，键入 xcodebuild 之后执行以下操作：

ARCHS=i386 GCC_TREAT_WARNINGS_AS_ERRORS=NO

其中的 ARCH=386 表示架构(ABI 的处理器型号)的二进制是目标。

### 2. iPhone 上 Objective-C

如果构建为 iPhone 代码，将需要根据是否持有一个苹果开发者证书来通过一个 SDK 选项。

如果有苹果开发者证书，通过 SDK 的 iPhoneOS。

如果没有苹果开发者证书，通过 SDK iphonesimulator。

### 3. 转换 Google Android 代码

SSR 提供了支持谷歌 Android 平台上运行的程序规则。这些规则：

- 可识别不安全的数据存储；
- 可进行安全权限分类，可检测 overprivileged 用途；
- 根据发送和接收的意图，识别数据库、文件系统、网络、私人信息和 Android 组件。

## 8.6.8 转换其他语言

### 1. 适用于其他语言的命令行语法

适用于其他语言的基本命令行语法是：

sourceanalyzer -b <build_id><file_list>

SQL 注释：默认情况下，在 Windows 平台上，会将扩展名为.sql 的文件视为 T-SQL，而不是 PL/SQL。如果正在使用 Windows 平台，并且具有扩展名为.sql 的 PL/SQL 文件，则应该配置 Fortify SCA 使其将这些文件视为 PL/SQL。要更改此默认行为，可将 fortify-sca.properties 中的 com.fortify.sca.fileextensions.sql 属性设置为 TSQL 或 PLSQL。

输入以下语句以便对 ColdFusion 源代码执行转换：

sourceanalyzer -b <build -id> -source-base-dir <dir><files|filespecifiers>

其中：

<build_id>指定项目的构建 ID

<dir>指定 Web 应用程序的根目录

<files|file specifiers>指定 CFML 源代码文件

ColdFusion 注释：Fortify SCA 将通过使用-source-base-dir 目录作为起始点，计算出每个 CFML 源文件的相对路径，然后在生成实例 ID 时使用这些相对路径。即便将整个应用源树移动到另一个目录，如果为-source-base-dir 指定了一个适当值，通过安全分析生成的实例 ID 也应该会保持不变。

表 8-4 显示出了各种文件说明符。

表 8-4　文件说明符

| 文件说明符 | 描　述 |
| --- | --- |
| <dirname> | 在指定目录或子目录下找到的所有文件 |
| <dirname>/**/Example.js | 在指定目录或子目录下找到的任何名为 Example.js 的文件 |
| <dirname>/*.js | 在指定目录下找到的任何带有扩展名.js 的文件 |
| <dirname>/**/*.js | 在指定目录或子目录下找到的任何带有扩展名.js 的文件 |
| <dirname>/**/* | 在指定目录或子目录下找到的所有文件(与<dirname>相同) |

注意：Windows 及许多 UNIX shell 会自动尝试展开含有'*' 字符的参数，因此应当使用文件说明符表达式。而且，在 Windows 中，输入反斜杠(\) 而不是正斜杠(/)。

**2. 其他语言命令行示例**

- 转换 PL/SQL 的示例

以下示例显示了转换两个 PL/SQL 文件的语法：

sourceanalyzer -b MyProject x.pks y.pks

以下示例显示了如何转换 sources 目录下的所有 PL/SQL 文件：

sourceanalyzer -b MyProject "sources/**/*.pks"

- 转换 T-SQL 的示例

以下示例显示了转换两个 T-SQL 文件的语法：

sourceanalyzer -b MyProject x.sql y.sql

以下示例显示了如何转换 sources 目录下的所有 T-SQL 文件：

sourceanalyzer -b MyProject “sources***.sql”

注意：此示例假定 fortify-sca.properties 文件中的 com.fortify.sca.fileextensions.sql 属性设置为"T-SQL"。

- 转换 PHP 的示例

要转换名为 MyPHP.php 的单个文件，输入：

sourceanalyzer -b mybuild "MyPHP.php"

- 转换用 VBScript 语言编写的 Classic ASP 的示例

要转换名为 MyASP.asp 的单个文件，输入：

sourceanalyzer -b mybuild "MyASP.asp"

- 转换 JavaScript 的示例

要转换 scripts 目录下的所有 JavaScript 文件，输入：

sourceanalyzer -b mybuild "scripts/*.js"

- 转换 VB 4.0 脚本文件的示例

要转换名为 myApp.vb 的 VB 4.0 文件，输入：

sourceanalyzer -b mybuild "myApp.vb"

- 转换 ColdFusion 的示例

以下示例显示了用于转换两个 CFML 文件的语法：

sourceanalyzer -b MyProject -source-base-dir .Page1.cfm Page2.cfm

以下示例显示了如何转换 C:\MySite 目录下的所有 CFML 文件：

sourceanalyzer -b MySite -source-base-dir C:\MySite "C:\MySite***.cfm"

## 8.6.9 故障排除与支持

**1. 故障排除**

1) 使用日志文件调试问题

如果在运行 Fortify SCA 时遇到警告或者问题，可使用-debug 选项再次运行 Fortify

SCA。这会在以下目录中生成一个名为 sca.log 的文件：

- 在 Windows 平台上：C:\Documents and Settings\<username>\LocalSettings\Application Data\Fortify\sca5.0\log
- 在其他平台上：$HOME/.fortify/sca5.0/log

将 sca.log 文件压缩为 zip 文件，通过电子邮件发送至 technicalsupport@fortify.com 以便进行进一步研究。

2) 转换失败的消息

如果 C/C++ 应用程序构建成功，但当使用 Fortify SCA 进行构建时却看到一条或多条“转换失败”消息，此时，请编辑<install_directory>/Core/config/fortify-sca.properties 文件，修改以下命令行：

com.fortify.sca.cpfe.options= --remove_unneeded_entities --suppress_vtbl

改为

com.fortify.sca.cpfe.options=-w --remove_unneeded_entities --suppress_vtbl

重新执行构建，输出转换器遇到的错误。如果输出的结果表明编译器与 Fortify 转换器之间存在冲突，可将输出发送给 Fortify 技术支持部门，以便进行进一步的研究。

3) JSP 转换问题

Fortify SCA 使用内置的或特定应用程序服务器的 JSP 编译器将 JSP 文件转换为 Java 文件以进行分析。如果在 Fortify SCA 将 JSP 文件转换为 Java 文件以进行分析的过程中，JSP 分解器遇到问题，会看到如下的消息：

Failed to translate the following jsps into analysis model. Please see the log file for any errors from the jsp parser and the user manual for hints on fixing those.
<List of JSP file names>

此问题通常由以下一个或多个原因造成：

- Web 应用程序的布局方式不是适当的可部署 WAR 目录格式；
- 缺少应用程序所需的某些 JAR 文件或类；
- 应用程序中缺少某些标签库或者其定义(TLD)。

要获取更多关于这个问题的信息，请执行以下步骤：

a) 在编辑器中打开 Fortify SCA 日志文件。

b) 搜索字符串 Jsp parser stdout:和 Jsp parser stderr:。

这些错误由所用的 JSP 分析器生成。请先解决这些错误，然后重新运行 Fortify SCA。

3) ASPX 转换问题

Fortify SCA 会将 ASPX 文件编译为 DLL 文件以进行分析，如下所示：

- 如果使用的是.NET 2.0(或更高版本)和 Visual Studio 2005，请使用 Microsoftaspnet_compile 编译器；
- 如果使用的是.NET 1.1 和 Visual Studio 2003，请设法每次从网站提取一个 ASPX 文件；

如果存在以下问题，编译操作就会失败：

- 存在访问或 authentication 问题，导致无法访问 Web 应用程序；
- 缺少某些必要的 DLL 文件。

在任何一种情况下，都会看到类似于以下内容的消息：

Failed to translate the following aspx files into analysis model. Please see the log file for any errors from the aspx precompiler and the user manual for hints on fixing those.

<List of ASPX file names>

如果正在使用插件，请启动插件调试，并检查插件日志文件中是否存在由 ASPX 预编译器生成的错误。

如果正在使用命令行工具 fortify_aspnet_compiler，应该会在控制台上看到错误消息。

如果仍无法确定问题的原因，请尝试从浏览器中访问失败的 ASPX 文件，看看会显示哪些问题。如果看到诸如 cannot locate assembly 之类的消息，请务必先获取缺少的 DLL 文件，然后重新运行 Fortify SCA。

如果能从浏览器中访问失败的 ASPX 文件，但 Fortify SCA 仍无法扫描该文件，请与 Fortify 技术支持部门联系以获得更多帮助。

4) C/C++预编译的头文件

一些 C/C++编译器支持一种叫做“预编译头文件”的功能，该功能可以加快编译。有些编译器实现这种功能后，会产生微妙的副作用。当激活这个功能时，编译器可能会不经过警告或者报错就接受错误的源代码。这种情况下会产生矛盾：即使编译器没有执行 Fortify SCA，也会报告转换错误。

如果使用编译器的预编译头文件功能，请确保源代码在禁用预编译头文件功能的情况下可以成功编译并执行完整的构建过程。

#### 2. 报告 Bug 和请求加强

反馈对于产品的成功必不可少。若请求加强功能或补丁，或者要报告 bug，请发送电子邮件至技术支持部门：technicalsupport@fortify.com。

请务必在电子邮件正文中包含下列信息：

产品：Fortify SCA

版本号：要确定版本号，请运行以下命令：sourceanalyzer -version

平台：(例如 PC)

操作系统：(例如 Windows 2000)

当请求加强功能时，请包含有关加强功能的描述。当报告 bug 时，请提供足够的详细信息以重现该问题。给出的描述越详尽，我们分析和修复问题的速度越快。同时，还要提供自问题发生以来的日志文件，或与问题相关的日志内容。

## 8.7 Audit Workbench 用户指南

### 8.7.1 Audit Workbench 简介

#### 1. Audit Workbench 概述

Audit Workbench 为 Fortify SCA 增加了一个图形用户界面，使得开发组织与安全审计小

组能够迅速地对分析结果进行组织、调查和划分优先级，从而在短时间内修复安全漏洞。

1) SCA 阶段概述

Audit Workbench 会启动 Fortify SCA"Scanning(扫描)"向导来扫描和分析源代码。该向导整合了以下几个分析阶段：

- 转换：使用源代码创建中间文件，源代码与一个 Build ID 相关联，Build ID 通常就是项目名称；
- 扫描与分析：扫描中间文件，分析代码，并将结果写入一个 Fortify Project Results (FPR) 文件；
- 校验：确保所有源文件均包含在扫描过程中，使用的是正确的规则包，且没有报告重大错误。

2) 安全编码规则包概述

安全编码规则包是 Fortify Software 安全研究小组多年软件安全经验的体现，并且经过其不断努力改进而成。这些规则是通过对编码理论和常用编码实践的研究，而积累的软件安全知识，并且在 Fortify Software 安全研究小组的努力下不断扩展和改进。每个安全编码规则包均包含大量规则，每个规则定义了一个被 SCA 检测出的特定异常行为。

一旦检测出安全问题，安全编码规则包会提供有关问题的信息，让开发人员能够有时间计划并实施修复工作，这样比研究问题的安全细节更有效。这些信息包括关于问题类别的具体信息、该问题会如何被攻击者利用，以及开发人员如何确保代码不受此漏洞的威胁。

安全编码规则包支持多种编程语言，也支持各种经过扩展的第三方库和配置文件。

### 2. Audit Workbench 界面概述

1) 开始页面概述

如图 8-32 所示，开始页面包含以下几个区域：

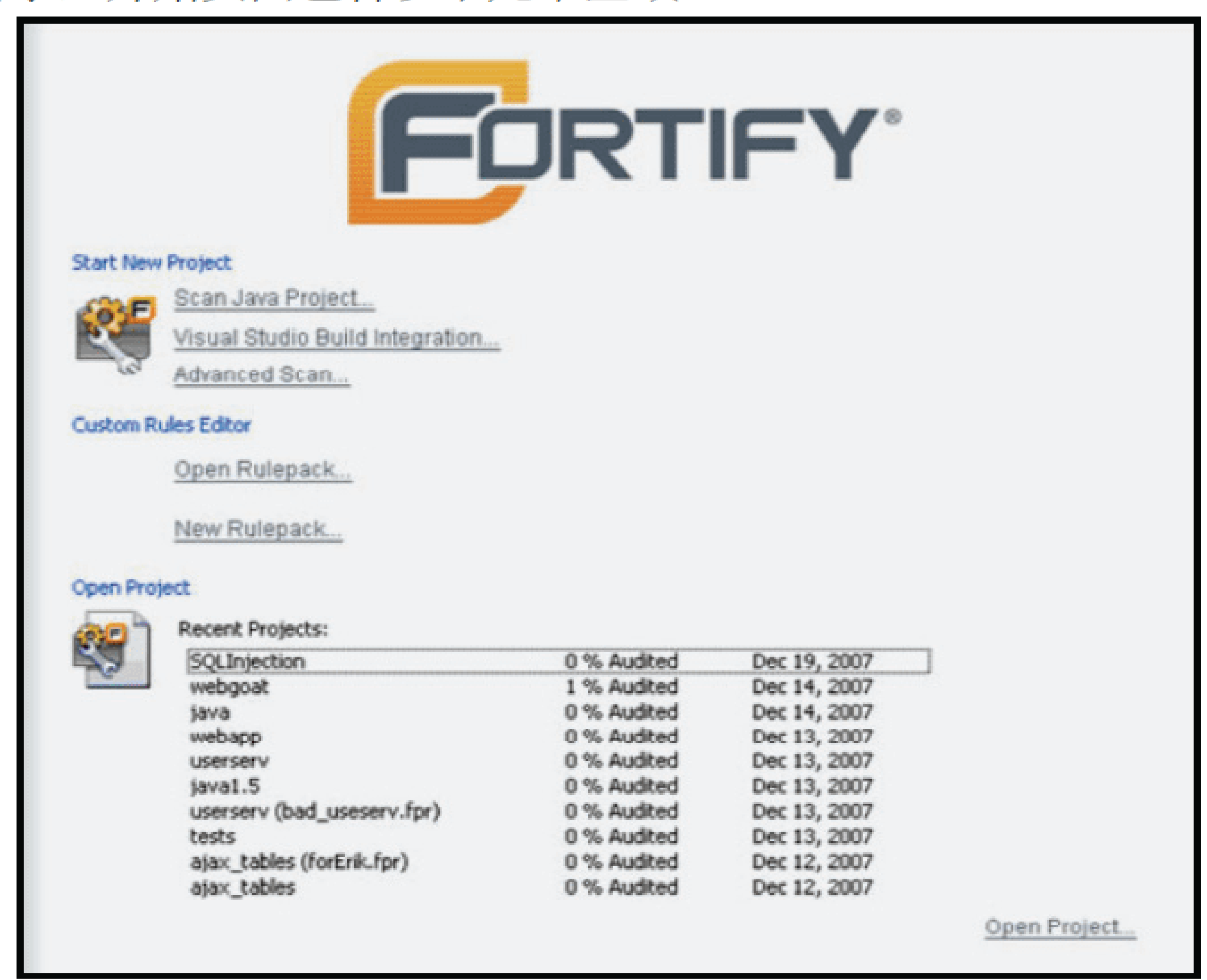



图 8-32　Audit Workbench 开始页面

- Start New Project(启动新项目)：启动源代码扫描向导；
- Custom Rules Editor (自定义规则编辑器)：启动用于创建和检查安全规则或编码实践

规则的工具，这些规则可针对源代码进行自定义；

- Open Project(打开项目)：单击某一项目名称可打开最近的项目；或使用“Open Project(打开项目)”链接可定位到某一项目。

2) 审计界面概述

图 8-33 显示了 Auditing (审计)界面中的 Webgoat FPR 文件示例。

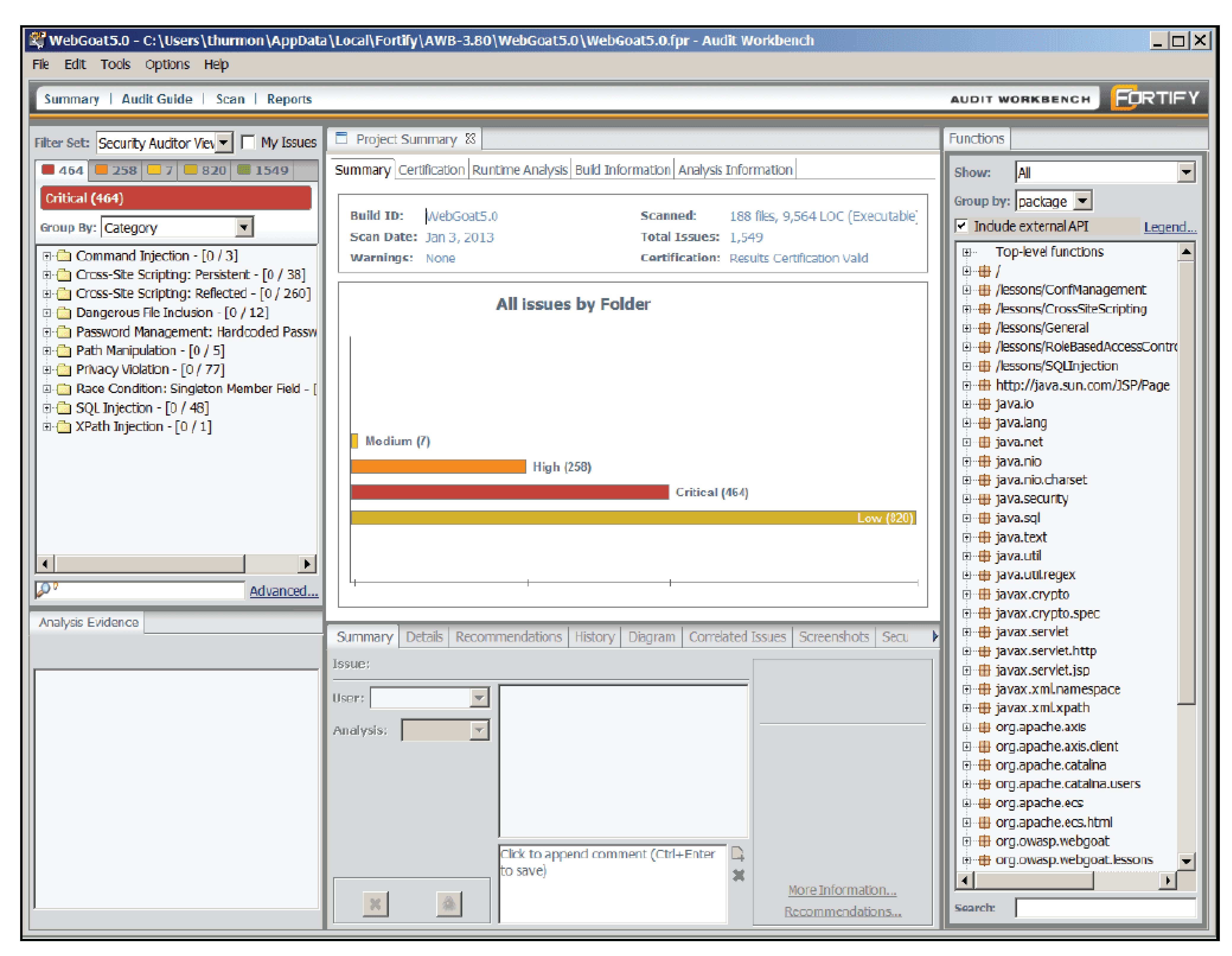



图 8-33　Audit Workbench 界面

Audit Workbench 界面由以下几个面板组成：

- Issues(问题)面板
- Analysis Trace(分析跟踪)面板
- Project Summary(项目摘要)面板
- Source Code Viewer(源代码查看器)面板
- Function(函数)面板
- Issue Auditing(问题审计)面板

Issues(问题)面板

如图 8-34 所示，使用 Issues (问题)面板，可以对希望审计的问题进行分组和选择。该面板由下列元素组成：

- Filter Set(过滤器组)下拉列表
- Folders(文件夹)选项卡
- Group by(分组方式)
- Search(搜索)框

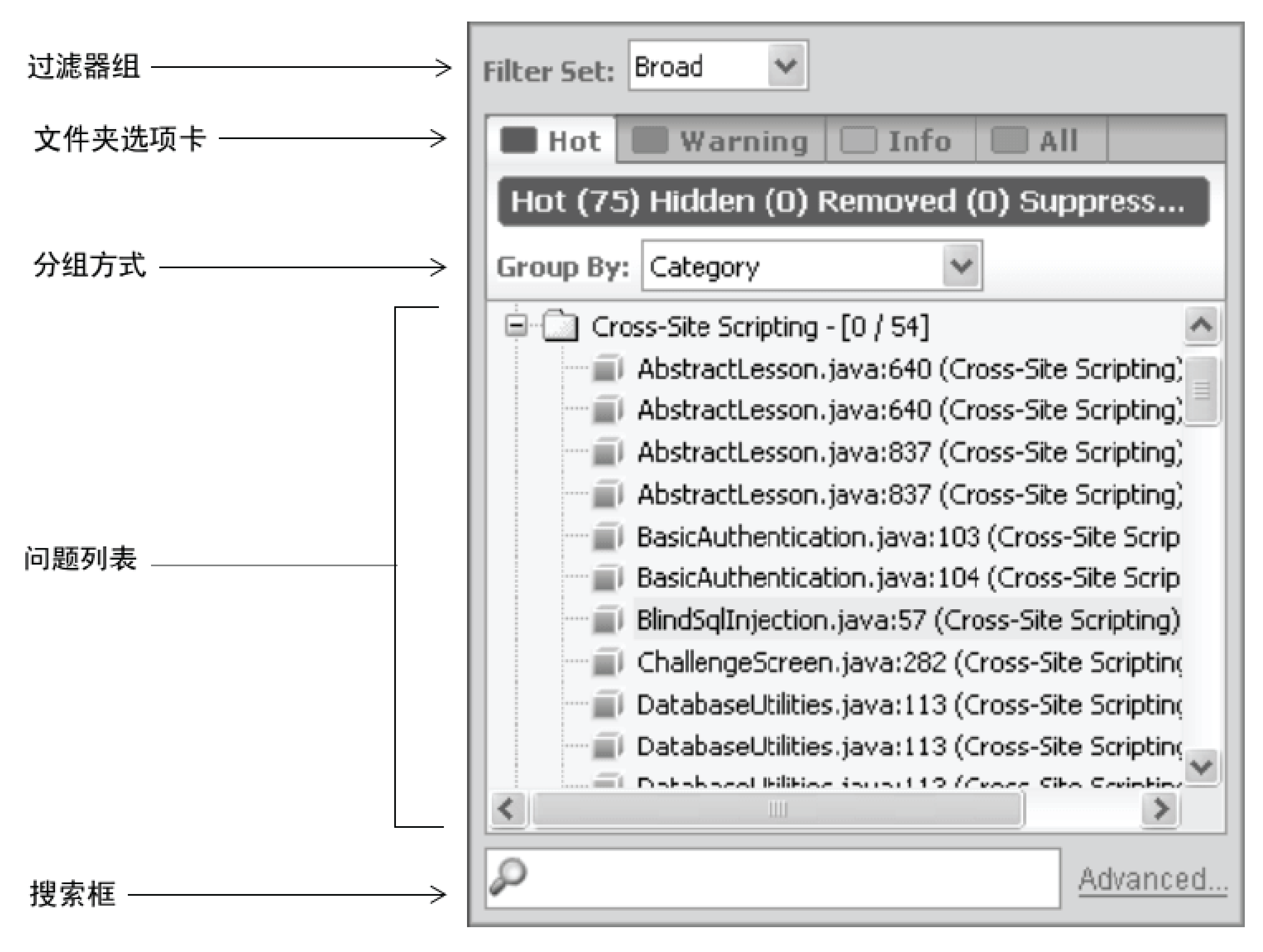



图 8-34　问题面板

Filter Set(过滤器组)下拉列表

过滤器组控制着 Issue(问题)面板的设置和显示属性。过滤器组是项目配置的一部分。可为每个项目自定义不同的过滤器组。每个项目均可具有唯一的过滤器组。

Filter Set(过滤器组)部分包含以下设置：

- 文件夹的名称和颜色。要进行配置，请选择 Tools(工具)，再选择 Project Configuration(项目配置)和 Folder Settings(文件夹设置)；
- 问题列出的位置，以及是否列出。要进行配置，请选择 Options(选项)，再选择 Show View (显示视图)和 Filters (过滤器)。

注意，Audit Guide Visibility Filters(审计指南可见性过滤器)配置适用于整个项目。

Fortify Software 提供了默认过滤器组：Broad(广泛)、Medium(普通)、Targeted(特定)和 Developer(开发人员)。Filter Sets(过滤器组)将问题按严重性归类于 Hot(严重)、Warning(警告)和 Info(信息)文件夹中。所有过滤器组都具有相同的文件夹过滤器。

不同“Filter Sets(过滤器组)”中会显示和隐藏不同的问题：

- Broad(广泛)：使用最全面的规则组显示问题。应使用该过滤器组来发现要审计的一组广泛的安全问题；
- Medium(普通)：显示了在产生结果(详述潜在问题)和产生预期可能的漏洞组之间打破平衡的问题；
- Targeted(特定)：显示在多种行业内及各种环境下经证实的具有高优先级的问题类别。应使用这些过滤器组来发现一组有限的众所周知的关键安全问题；
- Developer (开发人员)：显示开发人员尤为关注的问题，如高准确性 bug。

Folders(文件夹)选项卡

Issues(问题)面板上的选项卡称为“文件夹”。使用 Project Configuration(项目配置)，可以自定义文件夹及其设置。因此，不同过滤器组和项目之间的文件夹数目、名称、颜色和问题列表各不相同。

文件夹过滤器将问题归类于各种文件夹中。如果某个问题与任何文件夹过滤器都不匹配，该问题将被列在默认文件夹中。可见性过滤器决定了某个问题是否显示在列表中。

注意，可选择 Option(选项)、Show Suppressed(显示已废除项)、Show Hidden(显示隐藏项)(显示与可见性过滤器匹配的项目)以及 Show Removed(显示删除项)。

Group by(分组方式)

Group by(分组方式)选项用于将问题列表归类到各个子文件夹中。选定的选项会应用于所有可见文件夹。使用<none (无)>选项，可列出文件夹中的所有项目而不进行任何分组。

使用 Audit Workbench，通过更改对分组进行归类所依据的属性、添加或删除属性以创建子分组以及添加自己的分组选项，可对现有分组进行自定义。

Group by(分组方式)设置适用于 Audit Workbench 实例。可将分组方式选项应用于任何利用 Audit Workbench 实例打开的项目。

Search(搜索)框使用 Search(搜索)字段，可以限制文件夹中显示的问题和搜索特定的问题。

Analysis Trace(分析跟踪)面板

当选择某个问题后，Analysis Trace(分析跟踪)面板会显示相关的 trace output。通常情况下，这是一系列进程点，显示了分析器是如何找到该问题的。对于数据流和控制流问题，这一系列点会以执行顺序显示。

例如，如果选择某个与可能被感染数据流相关的问题，Analysis Trace(分析跟踪)面板会显示这段源代码中数据流的移动方向，如图 8-35 所示。

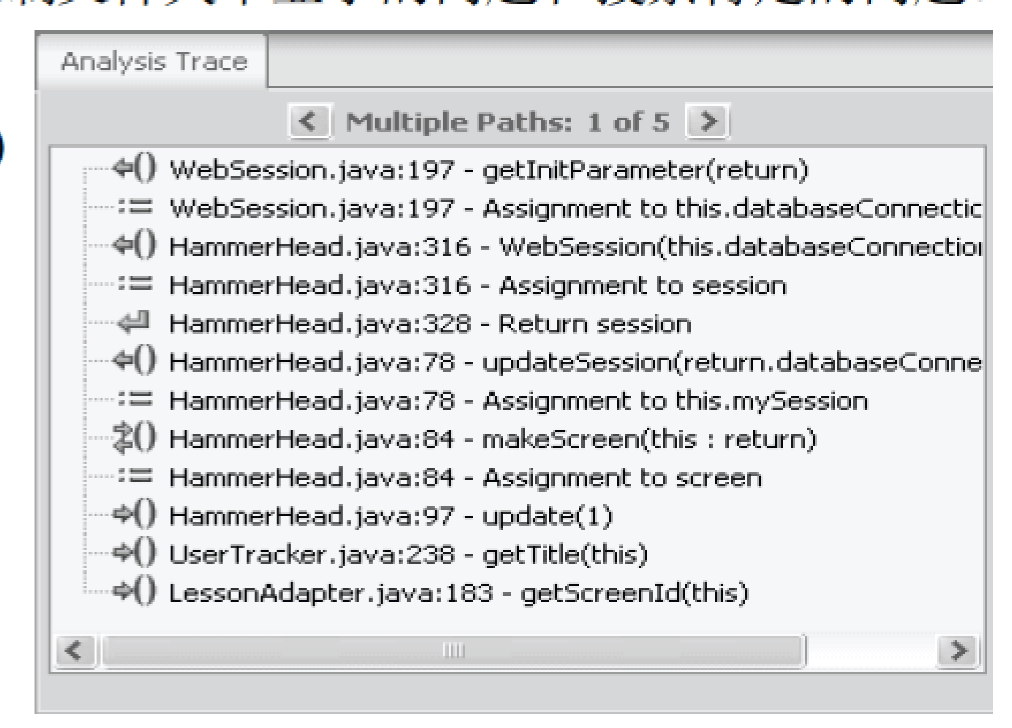



图 8-35　分析跟踪面板

Analysis Trace (分析跟踪)面板使用图 8-36 中的图标来显示本段源代码中数据流的移动方式。

| 图标 | 描述 |
|---|---|
| | 数据被分配到一个字段或变量 |
| | 信息读取自某个代码的外部数据源(如 HTML 格式和 URL 等) |
| | 数据被分配到一个全局字段或变量 |
| | 已执行比较 |
| | 函数调用收到被感染的数据 |
| | 函数调用返回被感染的数据 |
| | Pass-through，被感染的数据在函数调用中从一个参数传递到另一个参数 |
| | 为某个内存位置创建一个别名 |
| | 从一个变量读取数据 |
| | 从一个全局变量读取数据 |
| | 从一个函数返回被感染的数据 |
| | 创建一个指针 |
| | 取消一个指针 |
| | 变量范围终止 |
| | 执行跳跃 |
| | 在代码执行中引入分支 |
| | 代码执行中未引入分支 |
| | 常规 |

图 8-36　分析跟踪图标

Project Summary(项目摘要)面板

如图 8-37 所示，Project Summary(项目摘要)面板提供了关于扫描的详细信息。

- Summary(摘要)选项卡
- Certification(认证)选项卡
- Build Information(Build 信息)选项卡
- Analysis Information(分析信息)选项卡

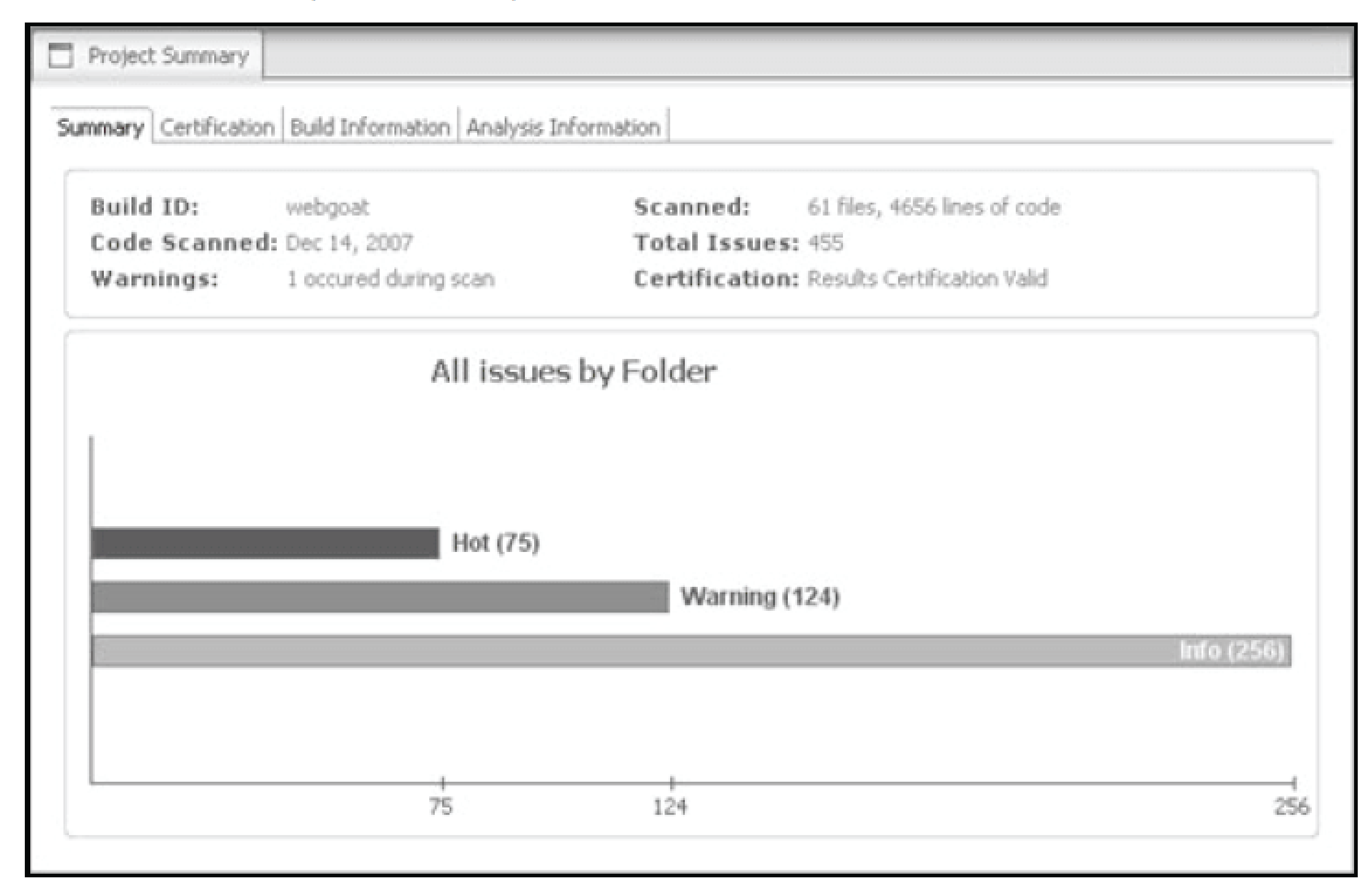



图 8-37　Project Summary 面板

Summary(摘要)选项卡

Summary(摘要)选项卡显示了关于本项目的高级信息。

Certification(认证)选项卡

Certification(认证)选项卡显示了结果认证状态。结果认证用于检查 FPR 文件是否与 FortifySCA 所生成的文件一致。

Build Information(Build 信息)选项卡

Build Information(Build 信息)选项卡显示了以下 Fortify SCA 扫描信息：

Build 详细信息，如 Build ID、扫描的文件数、扫描日期，可能会与文件的转换日期不同。

扫描的文件列表，包括文件大小和时间戳。

Java 类路径。

Analysis Information(分析信息)选项卡

Analysis Information(分析信息)选项卡显示了 Fortify SCA 的版本、计算机详细信息以及执行扫描的用户。

Analysis Information(分析信息)子选项卡包含以下信息：

规则包：列出用于扫描源代码的规则包，包括每个规则包的名称、ID 和版本号。

属性：显示 Fortify SCA 属性文件的设置。

命令行参数：显示用来分析项目的命令行选项。

警告：列出了分析期间发生的所有错误和警告。

3) Source Code Viewer(源代码查看器)面板

Source Code Viewer(源代码查看器)面板显示了与在 Issues (问题)面板中选择的问题相关

联的代码段。如果 Analysis Trace(分析跟踪)面板中的多个节点代表一个问题，则 SourceCode Viewer(源代码查看器)面板会显示与所选节点相关联的代码。

在 Source Code Viewer(源代码查看器)面板中，可使用代码辅助功能创建自定义规则和新问题，如下所示：

要为某个函数创建规则，可将光标放在该函数中，单击鼠标右键，然后选择 Write Rule for This Function(为此函数编写规则)。

要创建新问题，可将光标放在该函数中，单击鼠标右键，然后选择 Create New Issue(创建新问题)。

4) Function(函数)面板

如图 8-38 所示，Function(函数)面板显示了项目中的函数/方法列表。使用“Function (函数)”面板可找出该函数在源代码中的位置，了解函数是否应用和匹配某个规则，以及编写和验证自定义规则。




图 8-38　函数面板

Function(函数)面板具有以下选项：

- Show(显示)：确定显示在该列表中的函数；
- Group by(分组方式)：将函数归类到各个包和类中，显示层次结构或直接显示所有函数而不进行分组；
- Include unused functions(包括未使用的函数)：显示源代码中的每个函数。

默认情况下，列表中不会包含未使用的函数。

Legend(图例)：显示一个用于解释各个列表图标的窗口。函数旁边的图标颜色指明该函数是否应用了规则，如下所示：

红色：未应用任何规则。

蓝色：可应用规则，但没有匹配的规则。

绿色：可应用规则，并且有匹配的规则。

Search(搜索)：仅限于显示那些包含输入的字符串的函数。

该字段不区分大小写，如果该字段为空，将显示所选组中的所有函数。

右键单击某个函数可显示以下选项：

Open Declaration (打开声明)：显示该函数在源代码中声明的位置。

Find Usages (查找用法)：创建函数所应用于的文件位置的列表。会在“Audit (审计)”面板中打开“Search (搜索)”选项卡，其中显示它所在的文件名称。

Generate Rule for Function (为此函数生成规则)：启动“Custom Rules (自定义规则)”向导来编写将应用于该函数的规则。

Show Matched Rule (显示匹配的规则)：列出了与该函数相匹配的所有规则所属规则包的规则 ID 和文件名。

5) Issue Auditing (问题审计)面板

Issue Auditing (问题审计)面板在以下一组选项卡中提供了有关各问题的详细信息：

- Summary (摘要)
- Details (详细信息)
- Recommendations (建议)
- History (历史记录)
- Diagram (图示)
- Filter(过滤器)

注意：通过选择 Options(选项)，再选择 Show View(显示视图)菜单可显示或隐藏 Issue Auditing(问题审计)面板中的选项卡。

Summary(摘要)

如图 8-39 所示，Summary(摘要)选项卡显示了关于当前所选问题的以下信息。

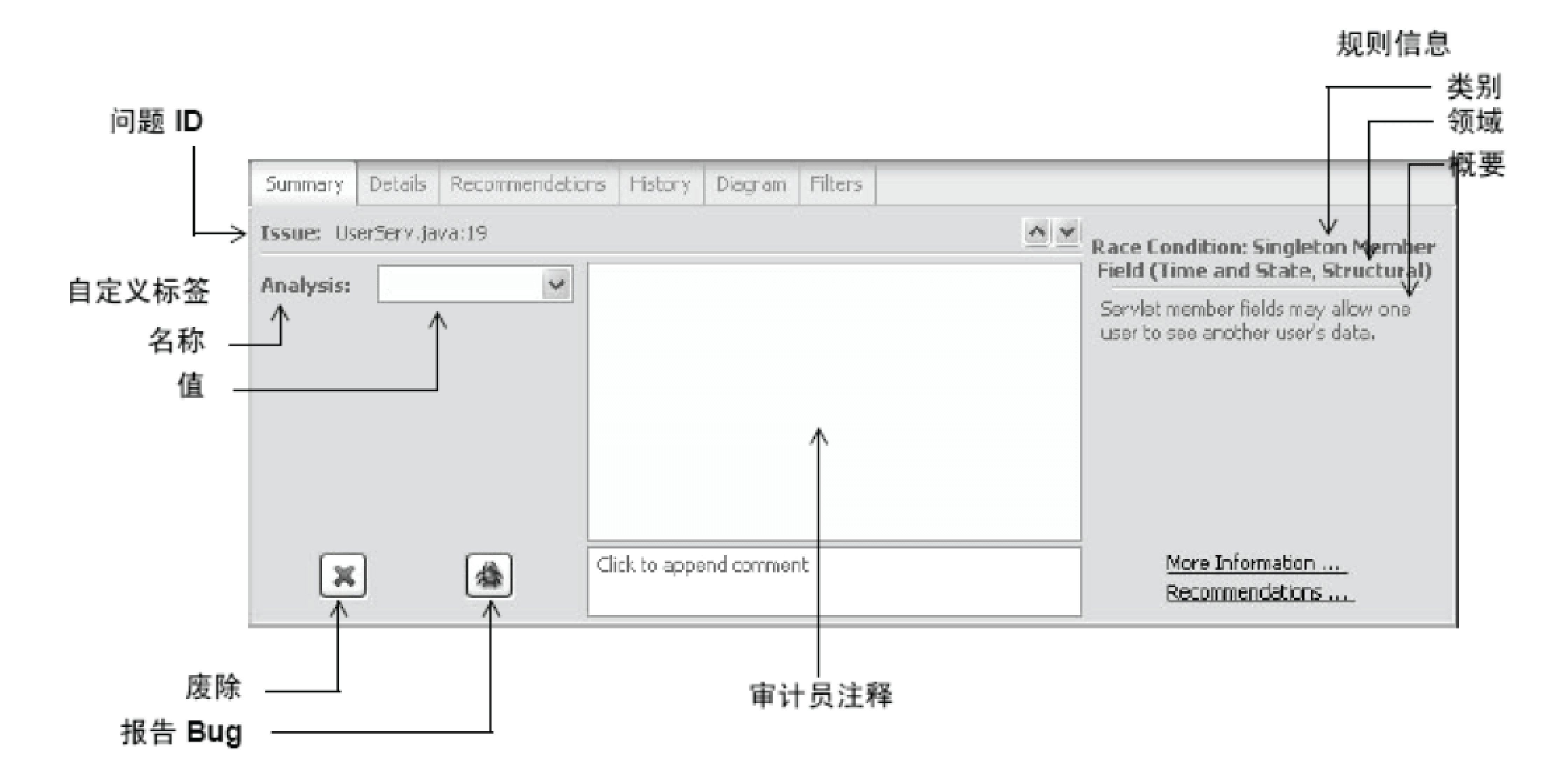



图 8-39　Issue Auditing 面板中的 Summary 选项卡

表 8-5 描述了 Summary(摘要)面板中的各个选项：

表 8-5　摘要选项

| 元　　素 | 描　　述 |
| --- | --- |
| 问题 | 显示问题的位置，包括文件名和行号 |
| Custom Tags<br>(自定义标签)区域 | 显示审计员可以作为问题属性添加的值的下拉列表。例如，默认情况下，“Analysis(分析)”标签提供了以下值：<br>- Not an issue(不是问题)<br>- Reliability issue(可靠性问题)<br>- Unknown(未知)<br>- Suspicious(可疑)<br>注意：自定义标签即项目设置，可选择“Tools(工具)，再选择 Project Configuration(项目配置)”来更改标签值 |
| Suppress Issue(废除问题) | 更改严重级别以废除和删除导航树中的问题 |
| File Bug (报告 Bug) | 提供访问 Bug 跟踪系统(如 Bugzilla)的途径 |
| Comments (注释) | 向注释字段中追加有关问题的其他信息 |
| 规则信息 | 显示用于描述问题的信息，如所属类别和领域 |
| 更多信息 | 打开 Details(详细信息)选项卡 |
| Recommendations(建议) | 打开 Recommendations(建议)选项卡 |

Details(详细信息)选项卡提供了有关所选问题的详细说明，并提供了用于解决该问题的指导方针。每项说明均包括表 8-6 中所述的部分或全部栏目。

表 8-6　详细信息面板

| 元　　素 | 描　　述 |
| --- | --- |
| Abstract (概要) | 对问题的概要描述 |
| Explanation (解释) | 关于会在哪些情况下发生此类问题的描述，同时还包括该漏洞的讨论、通常与该漏洞相关联的代码结构、该漏洞的利用方式以及潜在的衍生攻击类型 |
| 实例 ID | 问题的唯一识别符 |
| 规则 ID | 标识用于发现问题的主要规则 |
| SCA Confidence(SCA 可信度) | 一个由 Fortify SCA 生成的衡量指标 |
| + | 最大化选项卡 |

Recommendations (建议)选项卡包含有关如何避免引发该漏洞或修改该漏洞方法的建议与示例，如表 8-7 所示。

表 8-7　建议信息

| 元　　素 | 描　　述 |
| --- | --- |
| Recommendations(建议) | 关于如何解决问题的信息 |
| Tips (提示) | 提供普通或者难以审计的情况的示例，给出了有关如何识别特定问题的提示 |
| References(参考信息) | 参考相关信息 |
| + | 最大化选项卡 |

History (历史记录)显示完整的审计操作列表，其中包括以下详细信息：日期和时间、执行审计的计算机以及修正该问题的用户名称。

如图 8-40 所示，Diagram(图示)选项卡形象地说明了在 Issues(问题)面板上所选问题的节点执行顺序、调用深度以及表达式类型。

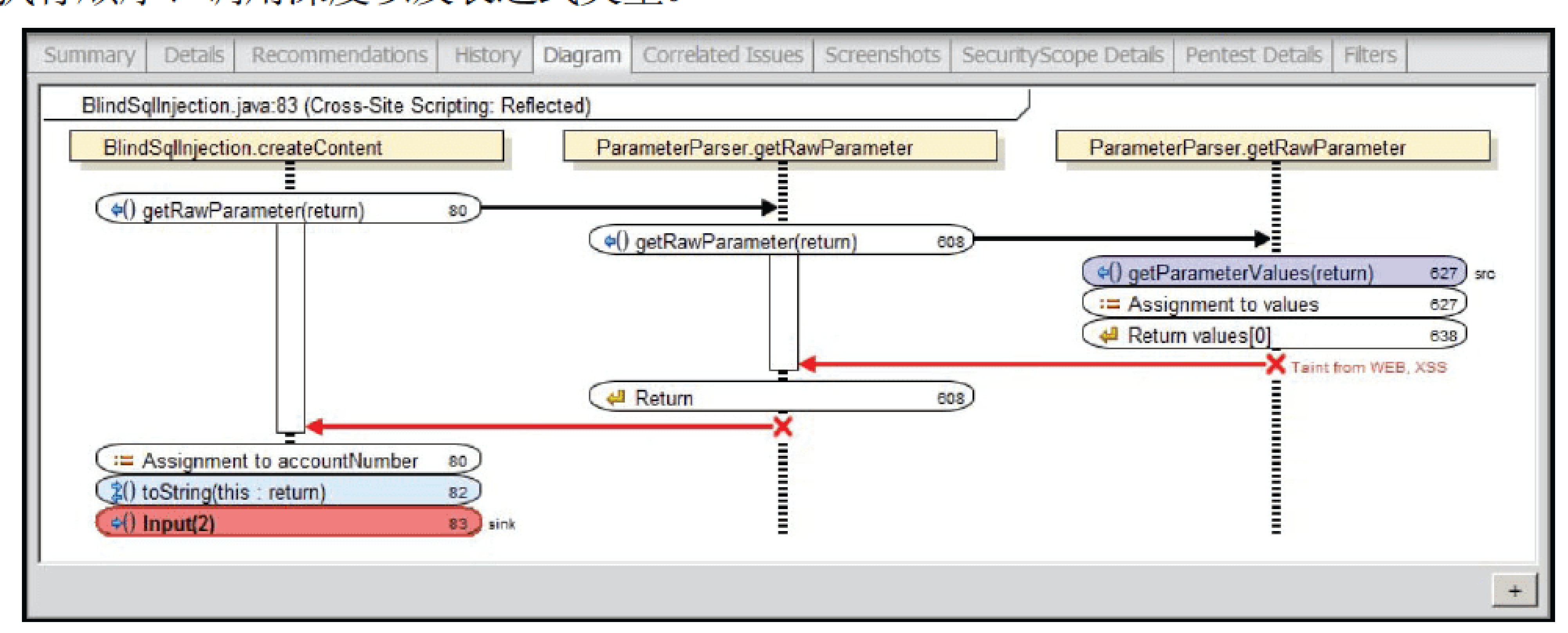



图 8-40　SQL Injection 问题数据流图示示例

Diagram(图示)选项卡显示了与规则类型相关的信息，如下所示：

数据流规则：纵轴显示执行顺序。跟踪会从顶部第一个调用感染源的函数开始，然后追踪对该 source(蓝色节点)的调用情况，并在 sink(红色节点)处结束跟踪。在图示中，也会标记 source(src)和 sink 结点。纵轴上的红 X 表示这个被调用的函数已结束执行。

横轴显示调用深度。用一条线显示控制权的传递方向。如果控制权传递时携带被感染数据的变量，则该线条是为红色；如果没有被感染数据，则该线条为黑色。

对于其他所有规则：纵轴显示执行顺序。每个节点的表达式类型用表 8-8 中的图标之一表示。

表 8-8　图示图标

| 图　标 | 描　述 |
|---|---|
| := | 数据被分配到一个字段或变量 |
| | 信息读取自某个代码的外部数据源(如 HTML 格式和 URL 等) |
| | 数据被分配到一个全局字段或变量 |
| | 已执行比较 |
| | 函数调用收到被感染的数据 |
| | 函数调用返回被感染的数据 |
| | Pass-through，被感染的数据在函数调用中从一个参数传递到另一个参数 |
| | 为某个内存位置创建一个别名 |
| | 从一个变量读取数据 |
| | 从一个全局变量读取数据 |
| | 从一个函数返回被感染的数据 |
| | 创建一个指针 |
| | 取消一个指针 |

(续表)

| 图　标 | 描　述 |
| --- | --- |
| | 变量范围终止 |
| | 执行跳跃 |
| | 在代码执行中引入分支 |
| | 代码执行中未引入分支 |
| | 常规 |

Filter(过滤器)

Filter(过滤器)选项卡显示了所选过滤器组中的所有过滤器，如表 8-9 所示。

表 8-9　过滤器选项卡中的选项

| 选　项 | 描　述 |
| --- | --- |
| Filter | 列出所选过滤器组中所配置的可见性过滤器和文件夹过滤器。右键单击某个过滤器可显示与该过滤器匹配的问题，也可启用或禁用该过滤器 |
| If | 显示过滤器条件。<br>第一个下拉列表列出了问题属性，第二个下拉列表指定了属性的匹配方式，而第三个列表是过滤器所匹配的值 |
| Then | 指示过滤器的类型，其中，hide 表示可见性过滤器，folder 表示文件夹过滤器 |

## 8.7.2　Audit Workbench 特性与功能

### 1. 扫描项目

1) 扫描 Java 项目

Scan Java Project(扫描 Java 项目)向导将转换阶段和分析阶段合并为一个简单步骤。使用此功能可扫描那些代码位于单个目录中的小型 Java 项目。

扫描新的 Java 项目：

a) 打开 Audit Workbench。

系统会显示开始页面。

b) 在 New Projects(新项目)中，单击 Scan Java Project(扫描 Java 项目)。

系统会显示 Browse for Folder (浏览文件夹)对话框。

c) 选择包含所有需要分析的源代码的文件夹，然后单击 OK(确定)。

注意：Fortify SCA 将 Build ID 设为文件夹名称。

系统会显示 AuditGuide Wizard (AuditGuide 向导)。

d) 为需要审计的问题类型选择相关设置，然后单击 Run Scan(运行扫描)。

如果Fortify SCA在扫描源代码时遇到任何问题，会显示Warning(警告)对话框。

e) 单击 OK (确定)继续。

Fortify SCA 会分析源代码。完成该过程后，Audit Workbench 会显示 FPR 文件。

2) 扫描其他项目

可使用 Advanced Scan(高级扫描)向导转换和分析 Java、JavaScript、PHP、ASP、.NET 和 SQL 项目。如果 Java 项目的源代码位于多个目录中、具有特殊转换或构建条件或包含需要从项目中排除的文件，应使用 Advanced Scan(高级扫描)向导。

扫描新项目：

a) 打开 Audit Workbench。

系统会显示开始页面。

b) 在 New Projects(新项目)中，单击 Advanced Scan(高级扫描)。

系统会显示Browse for Folder(浏览文件夹)对话框。

c) 选择项目所在的根目录，然后单击 OK(确定)。

会显示 Commandline Builder(命令行构建器)对话框，如图 8-41 所示。该向导会自动包含所有在扫描中支持的文件。

a) 另外，还可以添加其他目录中的文件：

单击 Add Directory(添加目录)。

系统会显示 Browse to Folder(浏览至文件夹)对话框。

选择包含希望添加进行扫描的文件所在的文件夹。

单击 OK(确定)。

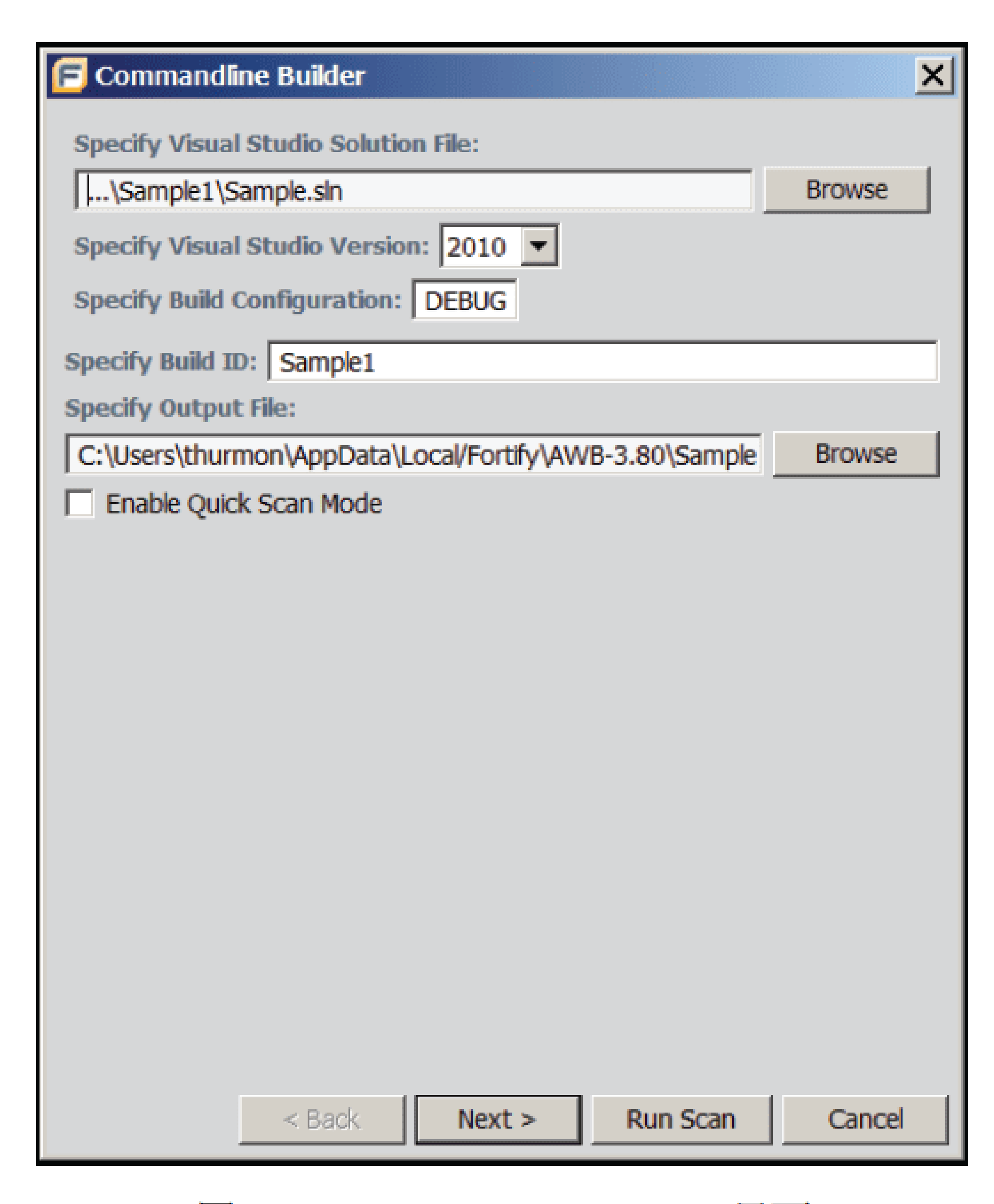



图 8-41　Commandline Builder 界面

导航面板中将会显示该目录，并会自动添加所有受支持的文件以进行扫描。要删除该目录，右键单击相应的文件夹，然后选择 Remove Root(删除根)。

b) 另外，可排除文件或目录，例如测试源代码：

选择所需文件或目录。

右键单击并选择 Exclude(排除)。

该文件夹会变成灰色。

c) 对于Java项目，设置以下内容：

选择包含类文件的文件夹，然后单击 Classpath Directory(类路径目录)。

在Fortify SCA转换阶段，该文件夹会变成蓝色，且文件会添加到类路径中。

选择项目的 Java 版本。

d) 输入Build ID。默认情况下，Build ID为根目录。

e) 输入Fortify SCA在分析阶段生成的FPR的路径和文件名。

f) 单击Next(下一步)。

系统会显示 Commandline Builder (命令行构建器)对话框，如图 8-42 所示。

要跳过某一阶段，请取消勾选 Enable Clean(启用清除)、Enable Translation(启用转换)或 Enable Scan(启用扫描)复选框。

根据需要，修改每个 Fortify SCA 阶段的命令行选项：

- Enable Clean(启用清除)：删除所有 Fortify SCA 与在 Select Source (选择源)对话框中设置的 Build ID 相关联的中间文件和内部版本记录。要更改 Build ID，可单击两次 Back(返回)；
- Enable Translation(启用转换)：创建 Fortify SCA 中间文件并为项目指定 Build ID；
- Enable Scan(启用扫描)：分析源代码并创建可在其中添加审计信息的 FPR 文件。

另外，还可单击 Manage Rulepacks(管理规则包)以使用自定义规则包分析源代码，或者禁用某个安全编码规则包。系统会显示审计指南向导，如图 8-43 所示。

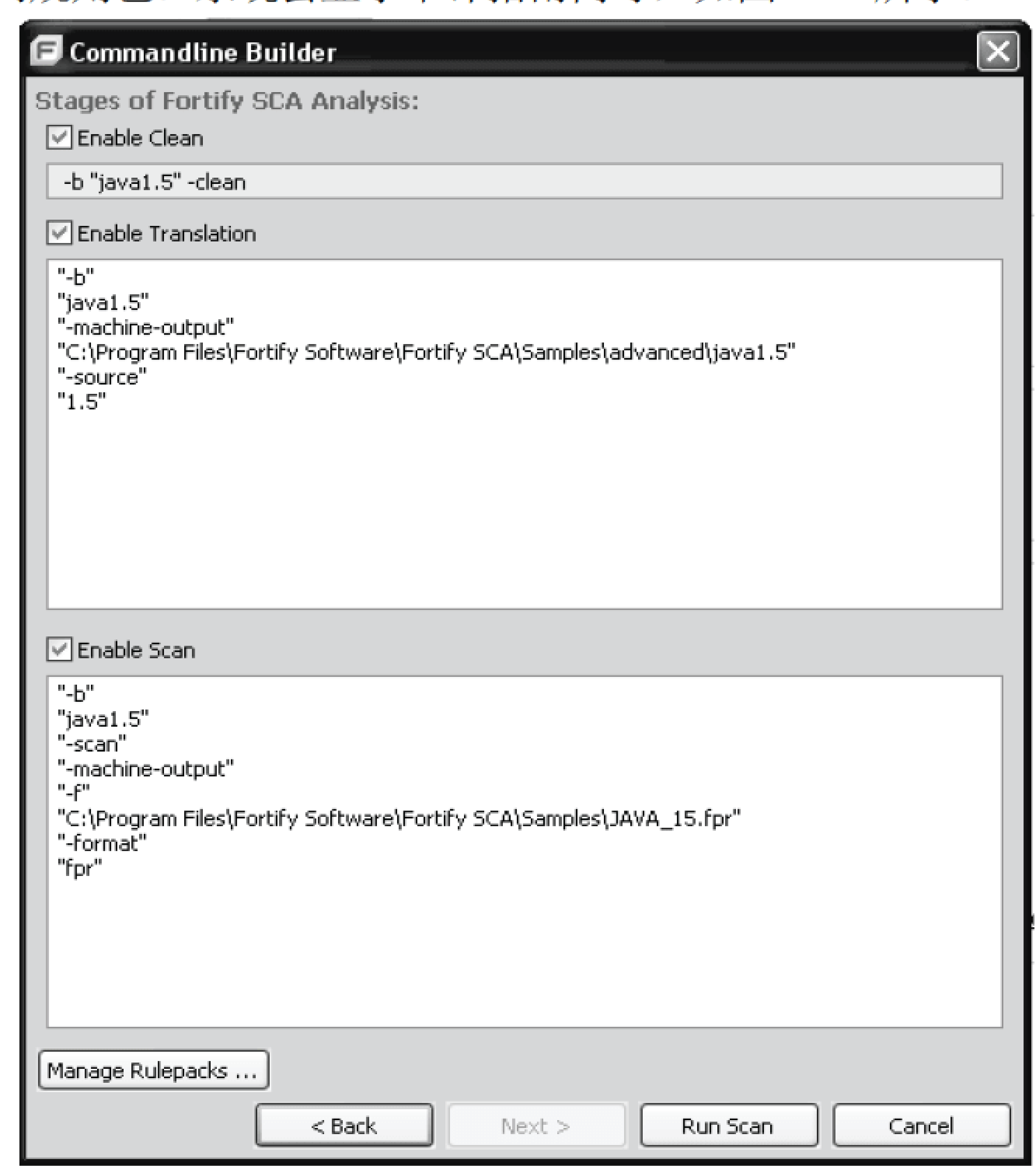



图 8-42　Commandline Builder 界面

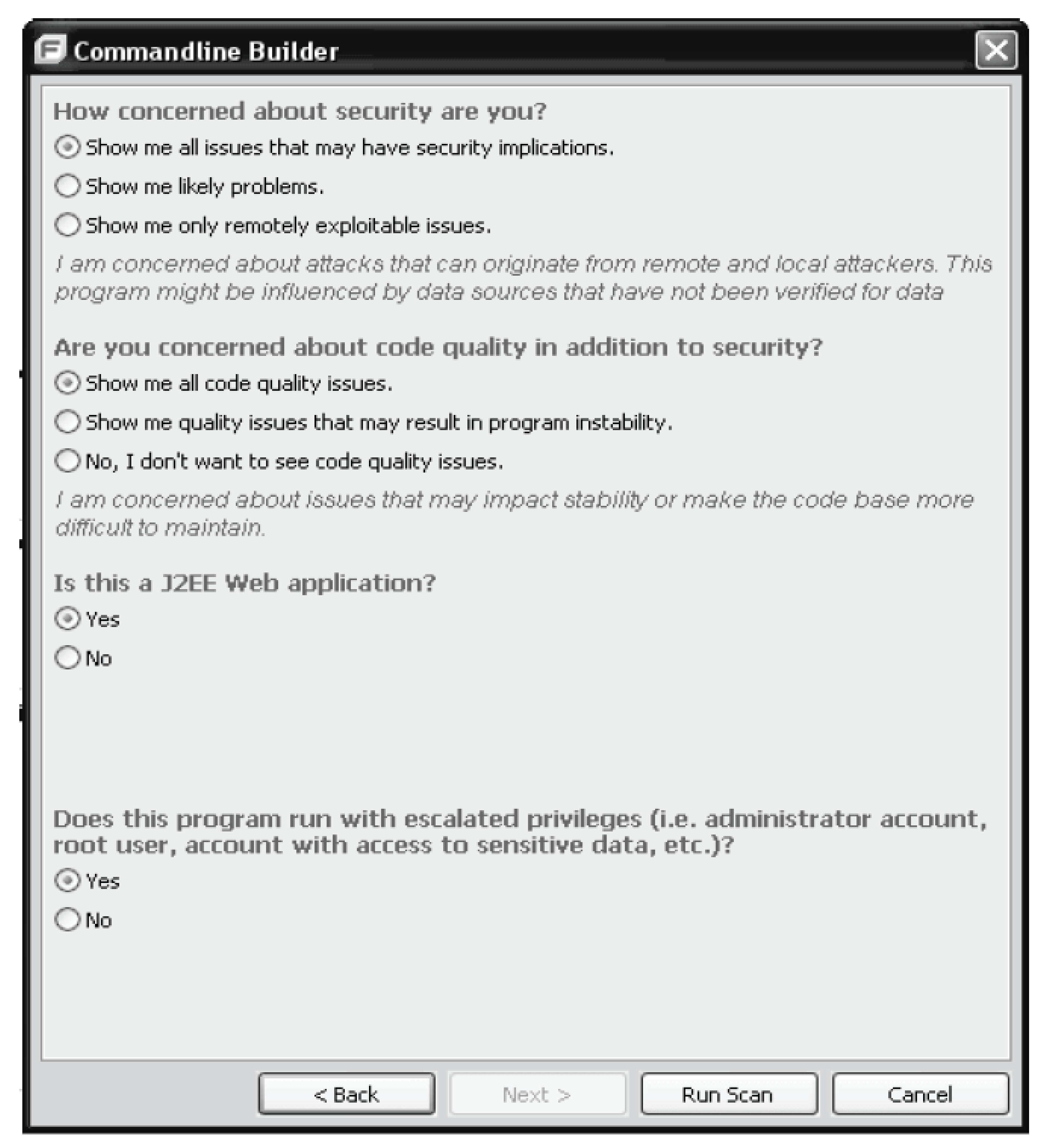



图 8-43　审计指南向导

g) 为希望进行审计的问题类型选择相关参数，然后单击 Next (下一步)。

h) 单击 Run Scan(运行扫描)。

如果 Fortify SCA 在扫描源代码时遇到任何问题，会显示 Warning(警告)对话框。

i) 单击 OK(确定)继续。

Audit Workbench 会显示 FPR 文件。

3) 重新扫描项目

a) 打开 FPR 文件。

b) 单击 Scan(扫描)。

系统会显示 Rescan Build ID (重新扫描 Build ID)对话框。

(可选)单击 Update Project Translation(更新项目转换)以重新转换该项目。

(可选)修改 Fortify SCA 扫描阶段的命令行选项。

(可选)单击 Manage Rulepacks(管理规则包)可更改用来分析该项目的规则包。

通过选中或清除复选框来添加或删除规则包。

要使用未列出的自定义规则包，单击 Add Custom Rulepack (添加自定义规则包)，浏览到规则包文件并选中，然后单击 OK(确定)。

c) 单击 Run Scan(运行扫描)。

扫描完成后，将显示 FPR。

d) 将新结果与之前 FPR 中的问题进行比较，如下所示：

要显示所有新问题，单击 All (全部)文件夹选项卡，然后选择 Group by New Issue(按新问题分组)选项。

要显示已删除的问题，单击 All (全部)文件夹选项卡，然后选择 Options(选项)，再选择 Show Removed Items(显示删除项)。

要检查本 FPR 中存在的、先前扫描所找到的项目，单击 All (全部)文件夹选项卡，然后展开 Existing Issues(现有问题)组。

4) 打开项目

打开 FPR 文件：

a) 打开 Audit Workbench。

b) 选择 File(文件)，再选择 Open Project(打开项目)。

c) 系统会显示 Choose Project(选择项目)窗口。

d) 浏览到 FPR 文件并选中，然后单击 Open(打开)。

如果 FPR 格式为 Fortify SCA 版本 4.5.1 或更早的版本，则会自动显示 Migration(迁移)向导。否则，FPR 会显示在 Audit(审计)透视图中。按照《Fortify SCA 迁移指南》中的说明，先迁移问题 ID，然后迁移审计数据。

### 2. 迁移审计信息

Audit Workbench 版本 5.0 改进了审计数据的管理方式。现在，已在用户界面中为每个项目基础自定义并配置了审计数据。打开项目时，会自动显示审计数据。若具有自定义的 Audit Labels (审计标签)，无须使用自定义审计标签信息修改 Audit Workbench 和 Fortify Manager 配置文件。每个项目都可具有一组唯一的标签。

在先前的版本中，审计员只能将预配置的信息添加到问题中。对于 Audit Workbench 实例而言，审计标签和值都是静态的、本地使用的。自定义 Audit Labels (审计标签)时，需要用户手动编辑 Audit Workbench 的配置文件。此外，还必须使用相同的自定义审计标签和值来手动重新配置每个 AuditWorkbench 和 Fortify Manager 实例。

打开版本 4.5.1 或更早版本的 FPR 文件时，会自动启动 Migration(迁移)向导。必须将旧的审计数据映射到新的 Custom Tags(自定义标签)特性。Fortify Software 建议先创建版本 5.0 的 Custom Tags(自定义标签)和 Filter Sets(过滤器组)，然后将旧的审计数据映射到新格式。

迁移审计信息：

1) 按照上一节的说明，打开 FPR 格式版本 4.5.1 或更早版本。

将自动保存旧的 FPR 备份。系统会显示 Migration(迁移)向导。

2) 另外，也可以选择一个迁移模板文件。

3) 单击 Next(下一步)。

4) 会显示 Migrate Analysis Values (迁移分析值)窗口，如图 8-44 所示。

5) 将旧的分析值映射到新值。

6) 另外，也可以选择 Filter Template(过滤器模板)。注意：只会导入分析标签的值。要使用过滤器、文件夹及其他设置，请在迁移审计信息之后重新导入模板。

7) 单击 Next(下一步)。

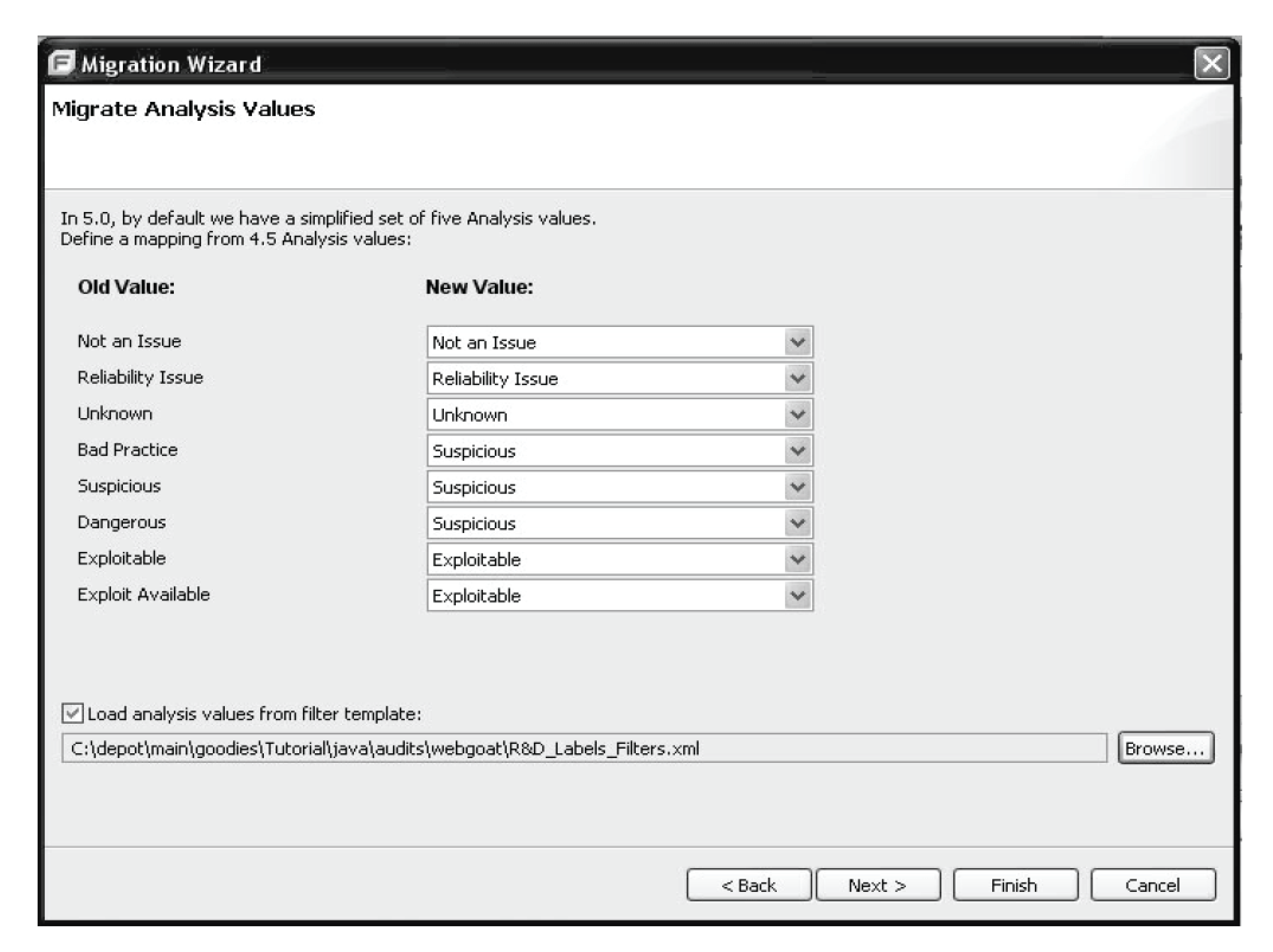



图 8-44 迁移分析值窗口

系统会显示 Migration Summary (迁移摘要)窗口，如图 8-45 所示。

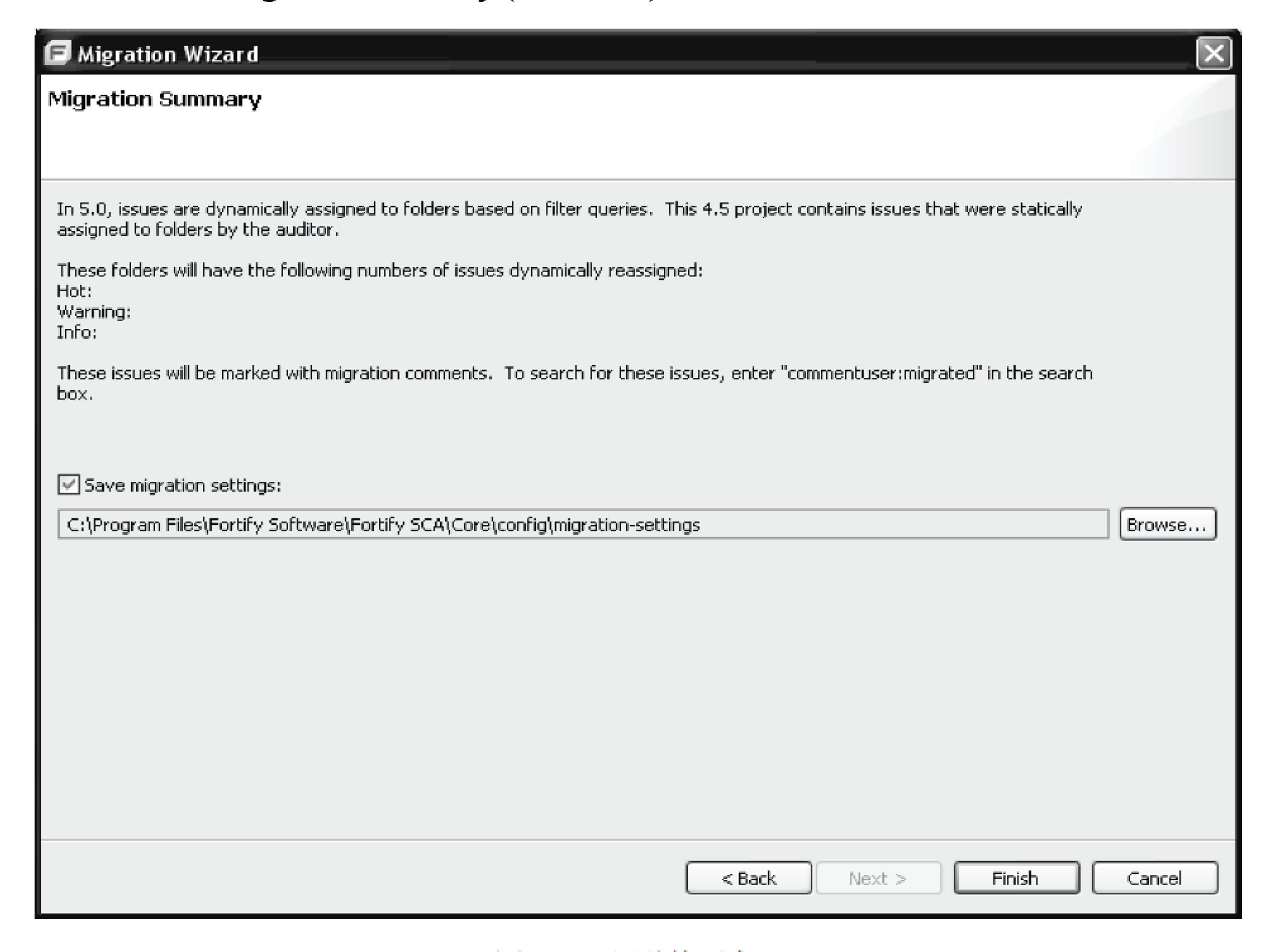



图 8-45 迁移摘要窗口

8) 单击 Finish(完成)。

### 8.7.3 配置/设置项目参数

#### 1. 理解项目配置和过滤器组

Fortify SCA 使用安全编码规则包和自定义规则包来分析源代码，确定其中是否包含任何安全漏洞或不安全的编程方式。Fortify SCA 会识别每个潜在的漏洞，并将其视为一个问题。然后 Fortify SCA 会创建一个 FPR 文件，其中包含关于该项目中存在的全部问题的详细信息，包括源代码片段和位置的详情(如文件名、包和类)。一个 FPR 文件可能包含许多问题。

过滤器组可控制Audit Workbench如何显示这些问题，而且允许用户自定义以下项目设置：

Folders (文件夹)，即 Issues (问题)面板上的选项卡：

- 显示的文件夹(选项卡)数量
- 每个文件夹的名称和属性

Custom Tags (自定义标签)，即 Issue Summary (问题摘要)选项卡上的审计标签：

- 可用审计字段的数量
- 每个字段的名称和值

每个过滤器组的过滤器：

- Visibility Filters(可见性过滤器)，可隐藏问题
- Folder Filters(文件夹过滤器)，可将问题分类整理到 Issues(问题)面板上的各个文件夹中

注意：版本 4.5 及更低版本中的 Analysis and status (分析和状态)等审计标签已经替换为 Custom Tags(自定义标签)。

#### 2. 保存项目配置参数

要保存项目配置参数，请选择 Fortify Software→Save Audit Project (保存审计项目)。

#### 3. 管理文件夹

Issues(问题)面板上的选项卡称为“文件夹”。可为每个项目自定义文件夹。文件夹在该项目内有效，可供该项目中每个过滤器组使用。在一个项目中，不同的过滤器组之间的文件夹属性可能会有所不同。

新建文件夹步骤：

1) 选择 Fortify Software→Project Configuration(项目配置)。屏幕中将显示 Project Configuration (项目配置)对话框。

2) 单击 Folder(文件夹)选项卡。

3) 从 Folders for Filter Set (过滤器组的文件夹)中，选择下列选项之一：

- All Folders (所有文件夹)，创建可在每个过滤器组中显示的文件夹；
- 一个过滤器组，创建的文件夹只能在所选过滤器组中显示。

4) 添加文件夹：

a) 单击 Folders (文件夹)旁的+(加号)。

屏幕上将显示 Create a New Folder(新建文件夹)对话框。

b) 为新文件夹输入一个名称，然后单击 OK(确定)。该文件夹将显示在文件夹列表的底部。

(可选操作)向上或向下拖动该文件夹，以改变该选项卡在 Issue(问题)面板上的位置，在面板中会按照从左到右的顺序排列。

(可选操作)选择 Default(默认)将所有与文件夹过滤器不符的问题放到此文件夹中。

单击OK(确定)。

该文件夹会在 Issue(问题)面板上显示为一个选项卡。如果选择了 Default(默认)，会显示所有与文件夹过滤器不符的问题。

注意：要将问题导入此文件夹，请添加一个以新文件夹作为目标文件夹过滤器。

添加文件夹步骤：

1) 选择 Fortify Software→Project Configuration(项目配置)。

屏幕中将显示 Project Configuration (项目配置)对话框。

2) 单击 Folder(文件夹)选项卡。

3) 从Folders for Filter Set(过滤器组的文件夹)中，选择要添加该过滤器的过滤器组。

注意：选择 All Folders(所有文件夹)选项，创建的文件夹可在每个过滤器组中显示。

4) 单击 Folders(文件夹)旁的+(加号)。

如果此过滤器组中有尚未显示的文件夹，则屏幕上将显示 Add a new Folder to the Filter Set(向过滤器组中添加新文件夹)窗口。

5) 选择要添加的文件夹，然后单击 Select(选择)。

该文件夹会显示在文件夹列表中。

6) 单击 OK(确定)。

该文件夹会显示在 Issue(问题)面板中，而且会选中修改过的过滤器组。

重命名文件夹步骤：

1) 选择 Fortify Software→Project Configuration(项目配置)。屏幕中将显示 Project Configuration (项目配置)对话框。

2) 单击 Folder(文件夹)选项卡。

3) 在列表中选择该文件夹。屏幕上将显示该文件夹的属性。

4) 为该文件夹输入一个新名称。在列表中，该文件夹的名称会随着键入而改变。

5) 单击OK(确定)。

在 Issue(问题)面板的选项卡上，该文件夹的名称也会随之改变。

删除文件夹步骤：

1) 选择 Fortify Software→Project Configuration(项目配置)。

屏幕中将显示 Project Configuration(项目配置)对话框。

2) 单击Folder(文件夹)选项卡。

3) 从 Folders for Filter Set (过滤器组的文件夹)中，选择下列选项之一：

- All Folders (所有文件夹)，将文件夹从所有过滤器组中删除。如果此文件夹是某个文件夹过滤器的目标，则该删除文件夹的选项会处于隐藏状态。
- 一个过滤器组，仅将该文件夹从所选过滤器组删除。文件夹列表会显示所选过滤器组中的文件夹。

4) 选择该文件夹，然后单击 Folders (文件夹)旁的 - (减号)。

如果该文件夹是某个文件夹过滤器的目标，则屏幕上将显示 Conflicts Occurred Removing a Folder(删除文件夹时出现冲突)窗口。

5) 根据需要重新设置文件夹过滤器的目标或删除该文件夹过滤器。

该文件夹便会从文件夹列表中消失。

6) 单击 OK(确定)。

该文件夹将不再在 Issue(问题)面板上显示为选项卡。

## 8.7.4　审计分析结果

### 1. 使用 Eclipse 导航和编辑特性

Audit Workbench 提供了源代码导航和编辑特性，这些特性的操作方式与 Eclipse 相同。可通过以下方式访问这些特性：

- Source(源)菜单，提供源代码的编辑和格式化工具；
- Search(搜索)菜单，提供关键字搜索及代码导航工具；
- Options(选项)菜单中的 Preferences(首选项)选项，允许自定义窗口和编辑器的显示方式。

### 2. 自定义问题视图

如图 8-46 所示，可以从 Options→Interface Preferences(选项→界面首选项)菜单中设置以下选项，以自定义 Issues(问题)视图：

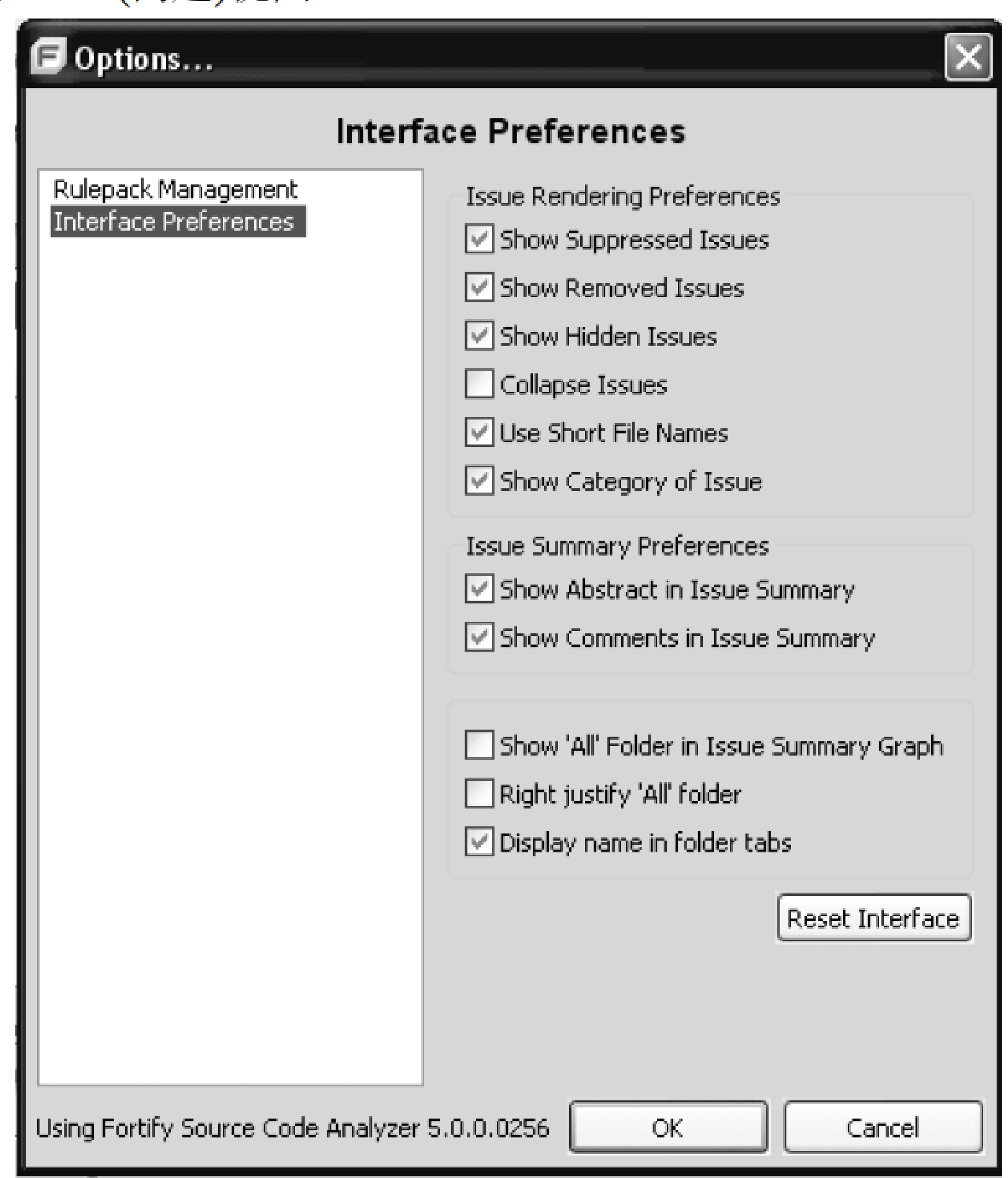



图 8-46　界面首选项窗口

- Show Entry Points (显示入口点)：显示用户输入进入程序的所有位置；
- Show Suppressed Items (显示废除项)：显示已废除的所有项。默认情况下，会禁用该选项；
- Show Removed Items (显示删除项)：如果已经执行了 Import New SCA Analysis(导入新 SCA 分析)，则会显示自上次分析以来已删除的所有项；
- Use Short Filenames(使用短文件名)：仅通过文件名而非相对路径来参照 Issues(问题)视图中的问题。默认情况下，已启用该选项；
- ShowView(显示视图)：允许显示 Analysis Trace(分析跟踪)、Issues(问题)以及 Summary(总结)面板。它还提供了 Other(其他)选项，其中包含标准的 Eclipse 导航特性。

### 3. 合并多个审计员的审计结果

Audit Workbench 允许多个审计员审计某个相同代码的各个部分，然后合并审计结果。要这样做，请选择 Tools(工具)，再选择 Merge Audits(合并审计)，然后导航到审计项目文件(.fpr) (其中包含要合并到当前项目中的审计后的问题)。

注意：仍需要进行外部协调，以确保不同的问题组都经过了审计。如果对相同的问题进行了两次审计，Audit Workbench 会发出警告，并允许执行以下其中一项操作：

1) 取消合并(未做更改)。

2) 对于有冲突的问题，使用当前审计员的更改。

### 4. 创建自定义清除规则

Audit Workbench 允许为特定的函数创建自定义清除规则。

创建清除规则：

1) 右键单击函数，打开关联菜单。

2) 选择 Write a Rule for This Function(为该函数编写规则)，打开 Custom Rules(自定义规则)向导。

3) 选择 Generic Validation Rule(一般验证规则)。

屏幕将显示 Select Language(选择语言)窗口。

4) 选择源代码的语言，然后单击 Next(下一步)。

5) 屏幕将显示 Function Information (函数信息)窗口。

6) 系统会自动植入所选函数的包、类和函数名。

7) 验证信息是否正确，然后单击 Next(下一步)。

屏幕将显示 Function Argument (函数参数)窗口。

8) 选择要清除的参数，然后单击 Next(下一步)。

屏幕将显示 Custom Rulepack Selection (自定义规则包选择)窗口。

9) 选择要在其中添加规则的规则包。

10) 单击 OK(确定)。

### 5. 从新的分析结果文件中更新

如果对某个分析结果文件执行了审计，随后对源文件重新运行扫描，则可以用新分析结

果文件中的结果来更新利用旧分析结果文件创建的审计项目文件，且不会丢失先前输入的审计信息。

利用新的扫描而创建的分析结果来更新审计项目文件：

1) 打开旧的审计项目文件。

2) 选择 Tools(工具)，再选择 Import New SCA Analysis (导入新的 SCA 分析)。屏幕将显示 Choose SCA Analysis File(选择 SCA 分析文件)窗口。

3) 选择新的分析结果文件。

4) 选择 File(文件)，再选择 Save As(另存为)，然后将新的分析结果文件另存为新的审计项目文件。

也可以使用 updatefpr.jar 命令行工具(可在不启动 Audit Workbench 的情况下执行更新)，利用新的分析结果文件更新审计项目文件。

updatefpr.jar 命令行工具位于以下目录中：

<install_directory>/Tools/updatefpr

使用语法如下：

java -jar updatefpr.jar <new>.fvdl old.fpr <new>.fpr

通常，会将审计数据从旧的扫描复制到新的扫描中，方法是使这两次扫描中的漏洞实例 ID 相匹配。

不过，由于分析引擎中发生更改，对于同一种逻辑漏洞，Fortify SCA 4.5 实例 ID 可能不同于 FortifySCA 4.0 实例 ID，从而导致某些审计数据的丢失。

对于同一版本的代码基数， migrate_audit_data 可将审计数据从 Fortify SCA 4.0 FPR 迁移到 Fortify SCA 4.5 FPR。即使实例 ID 各不相同，它也会查看漏洞结构以确定两种漏洞是否相同。

以下目录包含 migrate_audit_data 和 README.txt 文件(说明该工具的用法)：

<installation_directory>/Tools/migrate_audit_data

### 6. 导航并查看分析结果

在打开一个用来查看 Fortify SCA 所检测到的问题的 AuditWorkbench 项目后，可在 Summary (摘要)面板中审计这些问题，以反映这些问题的严重级别，以及对该问题执行的安全分析状态。

系统会按审计结果的严重性以及准确性，对审计结果进行分组，并在默认情况下将问题识别为位于 Issues(问题)面板中的 Hot(严重)列表内。单击 Warning(警告)和 Info(信息)，查看分组在这些列表中的问题。

每个列表中的问题数量显示在相关按钮下面。

要检验与 Issues(问题)面板中所列问题相关的代码，选中该问题。源代码部分包含显示在源代码查看器面板中的问题，并在面板的标题中显示文件所包含问题的名称。

### 8.7.5　生成报告

#### 1. 打开报告模板

1) 在 Audit Workbench 工具栏中单击 Reports (报告)。系统会显示 Generate Reports(生成报告)窗口。

2) 从 Report(报告)的下拉列表中选择报告模板。Generate Report(生成报告)窗口中会显示报告模板设置。

#### 2. 运行报告

选择所需报告模板和报告设置后，即可生成报告以查看其结果。可将报告结果另存为PDF、RTF 和 XML 文件。

1) 在 Audit Workbench 工具栏中单击 Reports(报告)。

系统会显示 Generate Reports(生成报告)窗口。

2) 从Report(报告)下拉菜单中选择报告模板。

3) 另外，还可更改报告一节的设置。

4) 单击Print Report(打印报告)。

会显示保存报告的对话框。

5) 指定文件名称和保存报告的位置。

6) 选择报告的文件类型。提供的选项有PDF、RTF或XML。

7) 单击Save(保存)。

已生成报告并另存为一个文件。

#### 3. 使用报告模板

首次安装 Fortify 产品后，下拉列表中会显示 Fortify 报告模板。如果用户已经编辑或创建了其他默认报告模板，可能无法查看上述默认报告模板。

Fortify 报告模板包括：

Fortify 安全报告：一种中级报告，可提供有关所执行分析的综合信息以及所执行审计的高级详细信息。还可为优先级最高的类别提供高级说明和示例。

Fortify 开发人员工作手册：一份综合列表，其中列出了已找到的所有问题类型，并为每种问题列举了多个示例。它还为每种类别的问题数目提供了高级汇总信息。

可选择要包含在报告中的各个小节，还可以编辑各小节显示的内容。

选择要包含在报告中的各个小节：

1) 在左侧列表中勾选每个小节标题的复选框，即可将其包含在报告中。

2) 单击小节标题可突出显示该标题。

窗口的右侧会显示小节的详细信息。

要从报告中删除某个小节，取消勾选小节标题旁的复选框即可。

选择小节标题后，便可以编辑在报告中显示的内容。可以编辑文本、添加或更改文本变量，还可自定义图标或结果列表中显示的问题。

其中包括：

- 编辑文本子小节
- 编辑结果列表子小节
- 编辑图表子小节

编辑文本子小节：

1) 勾选所需子小节标题旁边的复选框，即可将该文本包含在报告中。子小节标题下方会显示对文本的说明。

2) 单击 Edit Text(编辑文本)。文本框中会显示即将会包含在报告中的文本和变量。

3) 根据需要编辑文本和文本变量。

编辑文本子小节后，可以插入在运行报告时定义的变量，如表 8-10 所示。

表 8-10　文本变量

| 变　　量 | 描　　述 |
| --- | --- |
| $AUDIT_GUIDE_SUMMARY$ | 列出通过回答 Audit Guide (审计指南)的问题所创建的过滤器 |
| $CLASSPATH_LISTING$ | 扫描过程中使用的 JAR 文件，每行一个相对路径 |
| $COMMANDLINE_ARGS$ | 命令行参数的完整列表(其格式与项目汇总的格式相同) |
| $FILE_LISTING$ | 已扫描文件的列表，每个文件的格式为<relative filepath> # Lines # kb <timestamp> |
| $FILTERSET_DETAILS$ | 当前过滤器组正在使用的过滤器列表 |
| $FILTERSET_NAME$ | 当前过滤器组的名称 |
| $FORTIFY_SCA_VERSION$ | Fortify SCA 版本 |
| $LIBDIR_LISTING$ | 扫描过程中指定的 libdirs，每行一个相对路径 |
| $LOC$ | 代码总行数 |
| $NUMBER_OF_FILES$ | 已扫描的文件总数 |
| $PROJECT_BUILD_LABEL$ | 项目的内部版本标签 |
| $PROJECT_NAME$ | Build ID |
| $PROPERTIES$ | 分析阶段设置的属性的完整列表(其格式与项目汇总的格式相同) |
| $RESULTS_CERTIFICATION$ | 带有每个文件的有效性列表的完整认证详情(请参见项目汇总) |
| $RESULTS_CERTIFICATION_SUMMARY$ | 用于描述认证的短句(其格式与项目汇总的格式相同) |
| $RULEPACKS$ | 分析过程中使用的规则包的完整列表(其格式与项目汇总的格式相同) |
| $SCAN_COMPUTER_ID$ | 执行扫描操作的计算机主机名 |
| $SCAN_DATE$ | 使用语言环境的默认格式化类型进行的分析的日期 |
| $SCAN_SUMMARY$ | 已扫描代码库的汇总格式为# files，　# lines of code |
| $SCAN_TIME$ | 分析阶段所用的时间 |
| $SCAN_USER$ | 执行扫描的用户名 |
| $SOURCE_BASE_PATH$ | 代码库的源库路径 |
| $TOTAL_FINDINGS$ | 找到的问题总数，不包括已废除或删除的问题 |
| $WARNINGS$ | 已出现的警告的完整列表(其格式与项目汇总的格式相同) |
| $WARNING_SUMMARY$ | 扫描过程中发现的警告次数 |

编辑结果列表子小节：

1) 勾选所需子小节标题旁的复选框，即可将该文本包含在报告中。子小节标题下方会显示对结果列表的说明。

2) 单击列有标题的问题可展开各个选项。

3) 选择将按其对结果列表进行分组的各种属性。如果选择按类别进行分组，用于类别的建议、摘要和解释也会包含在报告中。

4) 可以使用搜索功能细化该子小节中显示的各种问题。查询内容会显示在 Subsection(子小节)字段的 Refine Issues (细化问题)中。

5) 勾选或清除各组复选框中的 Limit number of Issues(有限数量的问题)。

6) 如果勾选了该复选框，请键入每个组中显示的问题数量。

编辑图表子小节：

1) 勾选所需子小节标题旁的复选框，即可将该文本包含在报告中。子小节标题下方会显示对图表的说明。

2) 选择按其对图表进行分组的各种属性。

3) 可使用搜索功能细化该子小节中显示的各种问题。

查询内容会显示在 Subsection(子小节)字段的 Refine Issues(细化问题)中。

4) 选择图表类型。图表类型包括表格、直方图和饼图。

### 4. 保存报告模板

可将当前报告设置另存为新的模板，以便日后选择该模板运行更多报告。

将设置另存为报告模板：

1) 在 Audit Workbench 工具栏中单击 Reports(报告)。

系统会显示 Generate Reports(生成报告)窗口。

2) 从 Report(报告)下拉菜单中选择该报告模板。

3) 更改报告小节和子小节的设置。

4) 单击 Save as New Template(另存为新模板)。

即可保存为新的报告模板。当从报告下拉菜单中选择该报告模板的名称时，该模板的报告设置便会显示在 Generate Report(生成报告)窗口中。

### 5. 保存对报告模板的更改

可以保存对报告模板的更改，这样，当前的新设置即会显示为默认设置。

将某个报告模板另存为默认的报告模板：

1) 在 Audit Workbench 工具栏中单击 Reports(报告)。系统会显示 Generate Reports(生成报告)窗口。

2) 从 Report(报告)下拉菜单中选择要另存为默认报告模板的模板。

3) 另外，还可以更改报告小节和子小节的设置。

4) 单击 Save Settings as Default(另存为默认设置)。

下次打开 Generate Reports(生成报告)窗口时，窗口中即会显示该报告模板的设置。该报告模板的名称显示在 Report(报告)下拉菜单的顶端。

### 6. 编辑报告模板 XML 文件

报告模板会另存为 XML 文件。可以编辑这些 XML 文件，以做出更改或创建新的报告模板文件。编辑 XML 文件时，可以选择要包含在报告模板中的各个小节和每个小节的内容。

报告模板 XML 文件的默认位置为：

<install_directory>/Core/config/reports

通过在该目录中指定路径或更改 header.png 和 footer.png，还可以自定义报告中使用的标识语。

可通过编辑 XML 文件来添加报告小节。在 XML 的结构中，ReportSection 标签可定义一个新的小节。在定义的新小节中，Title 标签定义了小节名称，其中必须至少包括一个 Subsection 标签，用于在报告中定义该小节的内容。以下是 Fortify 安全报告中 Results Outline 小节的 XML 代码：

```
<ReportSection enabled=“false” optionalSubsections=“true”>
    <Title>Results Outline</Title>
    <SubSection enabled=“true”>
        <Title>Overall number of results</Title>
        <Description>Results count</Description>
        <Text>The scan found $TOTAL_FINDINGS$ issues.</Text>
    </SubSection>
    <SubSection enabled=“true”>
        <Title>Vulnerability Examples by Category</Title>
        <Description>Results summary of the highest severity issues.
        Vulnerability examples are provided by category.</Description>
        <IssueListing limit=“1” listing=“true”>
            <Refinement>severity:(3.0，5.0] confidence:[4.0，5.0]</
            Refinement>
            <Chart chartType=“list”>
                <Axis>Category</Axis>
            </Chart>
        </IssueListing>
    </SubSection>
</ReportSection>
```

在上述示例中，Results Outline 小节包含两个子小节。第一个子小节是名为 Overall number of results 的文本子小节。第二个子小节是名为 Vulnerability Examples by Category 的结果列表。一个小节的内容可包含任何子小节的组合。

添加报告子小节：

1) 添加文本子小节

在文本子小节中，可包含 Title 标签、Description 标签和 Text 标签。在 Text 标签中，方便用户可以在生成报告之前编辑其中的内容，当然仍可以提供默认内容。以下是 Results Outline 小节中 Overall number of results 子小节的 XML 代码：

```
<SubSection enabled="true">
    <Title>Overall number of results</Title>
    <Description>Results count</Description>
    <Text>The scan found $TOTAL_FINDINGS$ issues.</Text>
</SubSection>
```

在上述示例中，该文本子小节的标题为 Overall number of results。用于介绍该文本用途的说明文字为 Results count。用户可在运行报告之前编辑 text 字段中的文字，该内容使用了一个名为$TOTAL_FINDINGS$的变量。

2) 添加结果列表子小节

在结果列表子小节中，可以包含 Title 标签、Description 标签和 IssueListing 标签。在 IssueListing 标签中，可以定义 limit 的默认内容，并将 listing 设为 true。即便用户可以在生成报告之前编辑 Refinement 标签的内容，仍可以在报告中包含带有默认语句或不带默认语句的标签。要生成结果列表，需将 Chart 标签的 chartType 属性设为 list。还可以定义 Axis 标签。以下是 Results Outline 小节中 Vulnerabilities Examples by Category 子小节的 XML 代码：

```
<SubSection enabled="true">
    <Title>Vulnerability Examples by Category</Title>
    <Description>Results summary of the highest severity issues.Vulnerability examples are
     provided by category.</Description>
    <IssueListing limit="1" listing="true">
       <Refinement>severity:(3.0, 5.0] confidence:[4.0, 5.0]</Refinement>
       <Chart chartType="list">
            <Axis>Category</Axis>
       </Chart>
    </IssueListing>
</SubSection>
```

在上述示例中，结果列表子小节的标题为 Vulnerability Examples by Category。用于介绍该文本用途的说明文字为 Results summary of the highest severity issues。Vulnerability examples are provided by category。该子小节将会为每种包含符合 severity:[3.0，5.0] confidence:[4.0，5.0] (即 Refinement 标签的值)语句的问题类别(即 Axis 标签的值)列出(listing=true)一个问题 (limit="1")。

3) 添加图表子小节

在图表子小节中，可以包含Title标签、Description标签和IssueListing标签。在IssueListing标签中，可定义limit的默认内容，并将listing设为false。即便用户可以在生成报告之前编辑Refinement标签的内容，仍可以在报告中包含带有默认语句或不带默认语句的标签。要生成一个饼图，需将Chart标签的chartType属性设为pie。提供的选项包括table、pie和bar。用户可在生成报告之前更改此设置。还可定义Axis标签。

以下代码显示了图表子小节的一个示例。

```
<SubSection enabled="true">
    <Title>New Issues</Title>
    <Description>A list of issues discovered since the previousanalysis</Description>
    <Text>The following issues have been discovered since the lastscan:</Text>
    <IssueListing limit="-1" listing="false">
        <Refinement />
        <Chart chartType="pie">
            <Axis>New Issue</Axis>
        </Chart>
    </IssueListing>
</SubSection>
```

在上述子小节中，图表(limit="-1" listing="false")包含名为 New Issues 的标题以及包含 The following issues have been discovered since the last scan 内容的文本小节。由于"Refinement"标签为空，该图表将会包含所有问题，并按照 New Issue(即 Axis 标签的值)的值对这些问题进行分组。该图表将显示为一个饼图(chartType="pie")。

## 8.7.6 编写自定义规则

### 1. Fortify SCA 规则概述

默认情况下，Fortify SCA 使用安装的安全编码规则包用来检查源代码，并定义一系列可能出现的问题，如可被攻击者利用的安全漏洞和不良的编程实践。安全编码规则包中的规则分析了受支持语言的核心和扩展的 API 包中的元素，并将分析结果记录在 Fortify SCA 中。每个问题的解释包含了对问题的描述和建议的解决方案，用以保护漏洞免受攻击和修复不良的编程实践。可通过创建自定义规则包来准确地分析特定的应用程序(其中包含有关源代码元素附加信息的规则)，验证专门的安全规则，以及细化 Fortify SCA 所报告的问题。

要编写有效的自定义规则，熟悉已知的安全漏洞类别和通常与它们相关的函数类型是非常重要的。深入理解各类经常出现在特定类型漏洞的函数有利于在编写自定义规则过程中能够瞄准与安全相关的函数。确定函数的安全性是一件非常复杂的事情，然而事实证明花费在学习函数类型与漏洞类别之间关系上的时间的确是有益的。必须识别与安全相关的函数，并检查它们的个体行为(通过检查源代码，或借助 API 文档)以此来确定能够体现各个函数具体行为和与之相关的漏洞类别的正确规则形式。一旦确定好了这种联系，使用自定义规则编辑器来创建规则就相对简单了。

### 2. 使用自定义规则包

1) 新建自定义规则包

创建一个自定义规则包：

a) 在 Audit(审计)窗口中，选择 File(文件)，再选择 Open Custom Rules Editor(打开自定义规则编辑器)。屏幕上将显示 Fortify Audit Workbench Custom Rules Editor(Fortify Audit Workbench 自定义规则编辑器)窗口。

b) 选择 File(文件)，再选择 New Rule Pack(新规则包)。屏幕上将显示 Create a New Folder (新建规则包)对话框。

c) 在文件名栏，输入规则包的名称。

d) 单击 Save(保存)。在 Table View (表格视图)中将显示一个新的自定义规则包。

2) 打开 Custom Rulepack(自定义规则包)

可以在不打开新规则的情况下将它添加到现有的规则包中。

打开一个现有的自定义规则包：

a) 在 Audit(审计)窗口中，选择 File(文件)，再选择 Open Custom Rules Editor(打开自定义规则编辑器)。屏幕上将显示 Fortify Audit Workbench Custom Rules Editor(Fortify Audit Workbench 自定义规则编辑器)窗口。

b) 选择 File(文件)，再选择 Open Rulepack (打开规则包)。屏幕上将显示 Choose Rules File

(选择规则文件)对话框。

c) 选择所需的规则包，然后单击 Open(打开)。在 Table View (表格视图)中将打开一个新的自定义规则包。

3) 配置规则包的细节

规则包信息可以定义自定义规则。默认的规则包名称是文件名。Audit Workbench、Fortify SCATeam Server 和 Fortify Manager 可以通过显示 Rulepack/Name(规则/名称)元素帮助用户定义特定的规则包。

设置自定义规则包的细节：

a) 在 Audit(审计)窗口中，选择 File(文件)，再选择 Open Rulepack(打开规则包)。屏幕上将显示 Choose Rules File(选择规则文件)对话框。

b) 选择所需的文件，然后单击 Open(打开)。在 Table View(表格视图)中将显示一个新的自定义规则。

c) 单击 XML View(XML 视图)选项卡。

设置规则包细节，如下所示：

在 Name (名称)元素中的 CDATA 的括号内输入定义规则包的名称。

在 Description (描述)元素中的 CDATA 的括号内输入对规则包的简短描述。

d) 单击 File(文件)，再单击 Save Rulepack(保存规则包)。可以更改规则包的名称和描述。规则包细节的示例：

```
<Name><![CDATA[myCompany's Security Rules]]></Name>
<Description><![CDATA[Enforces security guide lines.]]></Description>
```

4) 安装自定义规则包

默认情况下，Fortify SCA 通过使用自定义规则目录中的所有可以找得到的自定义规则包来分析代码。自定义规则目录的位置：

<install_directory>/Core/config/ssm.properties

创建一个自定义规则包：

a) 在 Audit(审计)窗口中，选择 Options(选项)，再选择 Options(选项)。屏幕上将显示 Options (选项)窗口。

b) 选择 Rulepack Management(规则包管理)。屏幕上将显示 Ruelpack Management(规则包管理)面板。

c) 单击 Import Rulepacks(导入规则包)。屏幕上将显示 Select Rulepack(选择规则包)对话框。

d) 选择所需的自定义规则包，然后单击 Open(打开)。自定义规则包的副本会保存在自定义规则目录中。

### 3. 使用自定义规则向导

通过自定义规则，可以分析第三方和专有的库，发现无法扫描到的源代码信息以及针对可能被安全编码规则包视为问题的元素编写相关的规则。

使用向导编写新规则：

1) 在 Audit(审计)窗口中，选择 File(文件)，再选择 Open Custom Rules Editor(打开自定义规则编辑器)。屏幕上将显示 Fortify Audit Workbench Custom Rules Editor(Fortify Audit

Workbench 自定义规则编辑器)窗口。

2) 选择 File(文件)，再选择 New Rule(新规则)。屏幕上会显示 Rules Wizard (规则向导)。

3) 选择所需的规则类型。向导会提示输入有关选定规则类型的各种信息。

4) 在每个窗口输入所需信息，然后单击Next(下一步)。屏幕上会显示 Successful Rule Generation (规则生成成功)窗口。

5) 按照如下方式将新规则添加到规则包：选择 New Rule will be inserted into an already open file(将新规则插入到已打开的文件)，然后选择所需的自定义规则包。选择 New Rule will be saved to(将新规则保存至)，单击 Browse(浏览)，然后选择所需的自定义规则包。

6) 选择 Show XML source for the new rule(新规则显示 XML 源代码)。

7) 单击 Finish(完成)。

8) 规则被添加到规则包中。自定义规则编辑器所显示的规则包中会包含以 XML 视图方式显示的新规则。

9) 选择 File(文件)，再选择 Save Rulepack(保存规则包)。规则被保存到规则包中。

### 4. 编辑和删除规则

编辑规则：

1) 选择需要编辑的规则。

2) 单击 Details(详细信息)。屏幕上会显示 Rule Details Editor(规则细节编辑器)。

3) 更改 Rule Parameters and Rule Description (规则参数和规则描述)面板所需的信息。

4) 单击OK(确定)。

删除规则：

1) 选择需要删除的规则。

2) 单击 Delete(删除)。屏幕上会出现一个对话框，询问是否删除。

3) 单击 Yes(是)删除该规则，或单击 No(否)取消删除。

### 5. 在 XML 视图中编写规则

通过自定义规则，可以分析第三方和专有的库，发现无法扫描使到的源代码信息以及针对可能被安全编码规则包视为问题的元素编写相关的规则。

插入 XML 模板

1) 在 Custom Rules Editor(自定义规则编辑器)中，打开自定义规则包。在 Fortify Audit Workbench Custom Rules Editor(Fortify Audit Workbench 自定义规则编辑器)窗口中会显示 Table View(表格视图)。

2) 单击 XML View(XML 视图)。

3) 右键单击 RuleDefinitions(规则定义)标签。

4) 选择 Insert Rule Template(插入规则模板) - <template_name>。template_name 适用于 <install_directory>/Core/config/ruletemplates/rules 中的所有模板。规则中会插入所有需要的元素。

5) 在属性和元素中输入规则信息完成规则。必填的元素和属性会以栏中的红色 X 标识。

6) 完成后，选择 File(文件)，再选择 Save Rulepack(保存规则包)。

创建自定义规则模板

自定义规则模板是带有常规规则定义组件的 XML 文件。可以修改现有的模板或创建自己的模板。

创建模板：

1) 在Custom Rules Editor (自定义规则编辑器)的XML视图中复制规则定义。

2) 将规则粘贴到文本文件中。

3) 用希望显示在 Insert Rule Template (插入规则模板)中的名称命名文件，并将其保存在下面的目录中：

<install_directory>/Core/config/ruletemplates/rules 文件名将显示在可用模板列表中。

### 6. 配置自定义规则元素

1) 定义规则信息

规则信息的识别也可以在其他过程实现，如 Project Summary(项目摘要)面板中的AuditWorkbench Audit Issue Details(审计问题细节)选项卡。设置语言是必需的，指定标签和指定注意事项是可选的。

a) 设置语言

规则是特定语言。必须设置语言属性。格式版本号属性指定了规则版本号。当前安全编码规则包版本号是 3.4。以下示例显示了用于 Java 的 3.4 版本的语义规则：

<SemanticRule formatVersion="3.4" language="java">

b) 添加标签

将在输出中包含允许输入文字的标签元素。比如，标签文字将显示在 Analysis Trace(分析跟踪)面板中。输入字串设置标签，如：

<Label>Company Rule<Label>

c) 添加笔记

使用笔记标签，可以输入任何相关信息或关于新规则的评论。笔记仅显示在 XML 视图中。如下例所示，在 CDATA 的括号中输入文本：

<Notes><![CDATA[Enforces corporate security policies.]]></Notes>

2) 定义自定义规则描述

可以写下新规则描述，可以在选择 Details and Recommendations (细节和建议)选择卡时获得该描述。描述提供关于决定漏洞的标准和修正漏洞的补救方案的信息。

a) 编写概要

在概要元素中输入对问题的描述，该描述被设计作为一个摘要提供给对分类(category)未必熟悉的审计员的。这个描述不应试图解释分类，而应提供一个对本身性质的尽可能多的提示。这段文本会在一些狭窄的空间出现，所以它不应超过 10 个单词。

如以下例子所示，概要元素中的文字必须使用 CDATA 括号：

```
<Abstract>
    <![CDATA[Enter a description of the issue that the rule matches.]]>
</Abstract>
```

b) 编写解释

解释元素中的文字显示在 Details(细节)选项卡中。包括对漏洞的深入讨论，与漏洞相伴

的典型构造，漏洞会如何被利用，和可能发动攻击的范畴。

如下例所示，概要元素中的文字必须使用 CDATA 括号：

```
<Explanation>
    <![CDATA[Enter a detailed explanation on vulnerability.]]>
</Explanation>
```

c) 编写建议

建议元素中的文字显示在 Recommendation(建议)选项卡中。该文字应解释如何修复漏洞。

```
<Recommendations>
    <![CDATA[Enter instructions on how to fix the issue.]]>
</Recommendations>
```

d) 编写提示

提示元素中的文字显示在 Recommendation(建议)选项卡中。包括普遍或困难的审计情况的范例，带有如何鉴别特定问题的小提示。随着在真实审计中新情况的发现，这个部分的内容应该逐渐增多。小提示应该是真实的，并对努力要搞清楚一份 source code analysis 结果的人有帮助。

```
<Tips>
    <![CDATA[Enter a tip.]]>
</Tips>
```

3) 定义领域、类别和严重性

当编写以下规则类型时，请设置领域(可选)、类别和严重性：

Configuration

Control Flow

Data Flow Sink

Semantic

Structural

a) 使用自定义规则向导

在自定义规则向导中，请选择以下的漏洞信息:

- 类别：选择一个类别或输入与此规则匹配的一个事件的新类别(最多 255 个字符)。列表显示出用于安装的自定义和安全编码规则包的规则的类别；
- 严重性：为与此规则相符合的事件设置 severity(严重)属性，可选择 Low(低)、Medium(中)、High(高)或 severity(严重)。

b) 使用 XML View (XML 视图)

在 XML View(XML 视图)中为表 8-11 中的标签设置规则信息值。

表 8-11　自定义规则信息字段

| 元　素 | 数据类型 | 描　述 |
|---|---|---|
| VulnCategory | 字符串 | 对漏洞的分类和分级，如 Buffer Overflow 或者 SQL Injection |
| DefaultSeverity | 浮点 | 显示漏洞重要性级别的范围，从 1.0 到 5.0 |
| VulnKingdom | 字符串 | 漏洞所属的领域 |
| VulnSubcategory | 字符串 | 漏洞类别的子类别 |

注意：按下 Ctrl+Space 会列出用于本地安装的安全编码规则包和自定义规则包的规则类型的值，如图 8-47 所示。




图 8-47　自定义规则向导：类别和严重性屏幕

3) 定义函数

a) 使用自定义规则向导

如图 8-48 所示，使用 Java 风格正则表达式以识别在 Function Information(函数信息)窗口中与所有规则类型相符的函数。以下将适用于除了配置与别名之外的所有规则类型中：

数据包：如果已定义将触发当前规则的函数，请输入与命名空间或数据包相符的正则表达式。

类别：如果已定义会触发当前规则的函数，请输入与类别相符的正则表达式。

函数：为将触发当前规则的函数或方法输入正则表达式。

注意：别名规则用于定义窗口中函数，将在下一节描述别名规则。

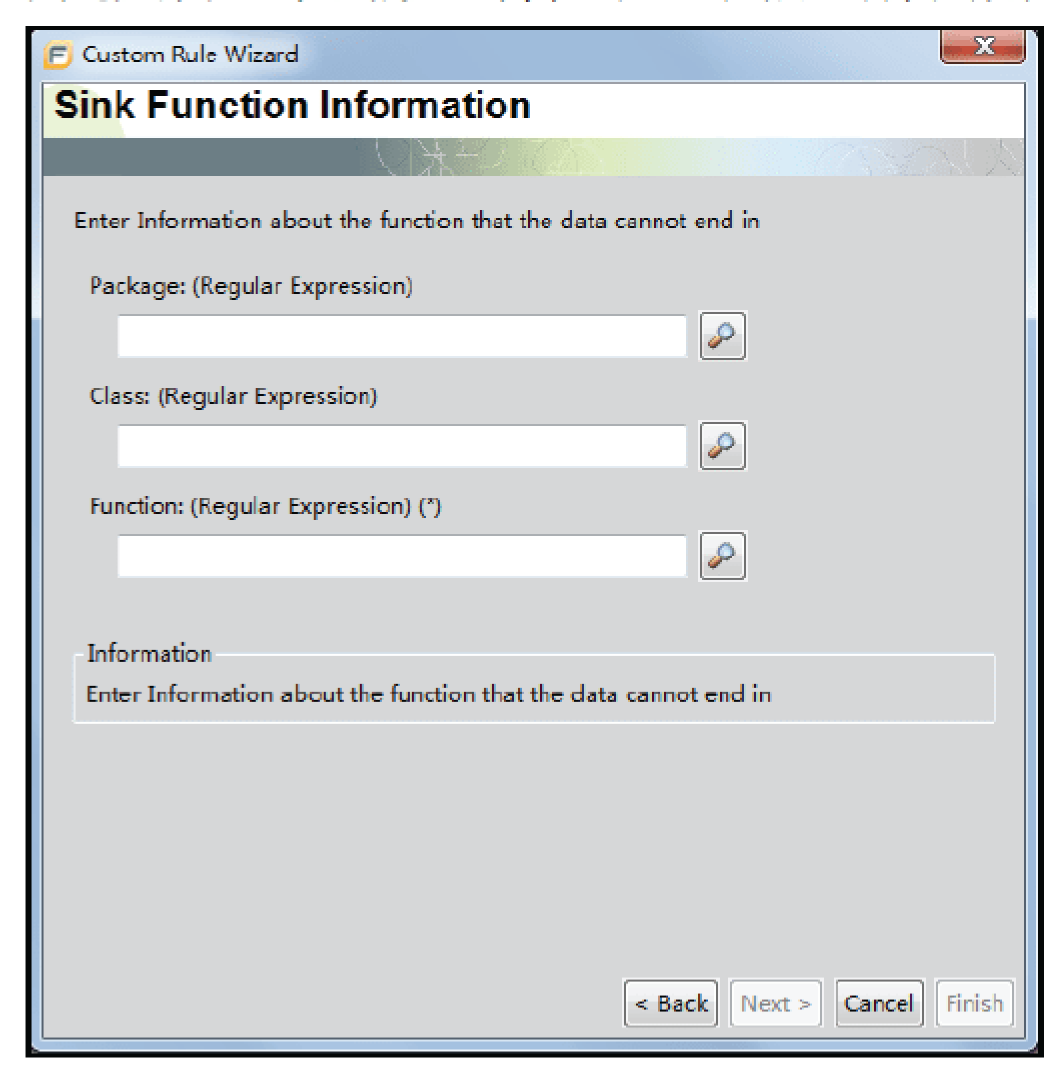



图 8-48　自定义规则向导：函数信息窗口

b) 使用 XML View (XML 视图)

定义命名空间、类别和函数

FunctionIdentifier 元素用于识别应用规则的函数，如表 8-12 所示。

表 8-12　识别 XML 中的函数

| 元素名称 | 内　容 | 类　型 | 描　述 |
|---|---|---|---|
| NamespaceName | TEXT | string | 如果已定义将触发当前规则的函数，请输入与命名空间或数据包精确匹配的字符串 |
| | Pattern | regex | 如果已定义将触发规则的函数，请输入与命名空间或数据包名称相符的正则表达式 |
| ClassName | TEXT | string | 如果已定义将触发规则的函数，请输入与类别精确匹配的字符串 |
| | Pattern | regex | 如果已定义会触发规则的函数，请输入与类别相符的正则表达式 |
| FunctionName | TEXT | string | 输入能够触发规则的函数或方法的名称 |
| | Pattern | regex | 输入与触发规则的函数名称相符的正则表达式 |

识别并定义函数参数

定义函数参数可以限制这样的规则，即与那些有其他特定值的函数相匹配并符合某一个签名(参数列表)的函数。限制函数基于签名触发规则在语言中十分有用，允许具有相同名字函数的定义能够重载，以便接受不同参数。

如表 8-13 所示，在参数元素中定义函数的参数：

表 8-13　函数的参数

| 元　素 | 属　性 | | | 描　述 |
|---|---|---|---|---|
| | 名　字 | 类　型 | 值 | |
| Parameters | varArg | Boolean | true \|false | 用于识别函数是否具有参数的变量数 |
| ParamType | | | | 字符串和那些在特定函数中定义的参数类型相对应。空的或未识别元素将指引 Fortify SCA 不受任何限制地与任何方法或函数签名相匹配 |
| WildCarda | min | int | | 整数代表了匹配的任意类型参数的最小数量，出现在特定函数的签名的最后 |
| | max | int | | 整数代表了匹配的任意类型参数的最小数量，出现在特定函数的签名的最后 |

## 7. 验证自定义规则

验证自定义规则包：创建简单的测试代码来验证所编写的规则，并将新的自定义规则保存到测试规则包中。使用 FortifySCA 仅根据新的自定义规则来扫描测试代码。

注意：验证规则时，通过选择 Options(选项)，然后选择 Show Hidden(显示隐藏)、Show Removed(显示删除)和 Show Suppressed (显示废除)来显示所有问题。

使用新的自定义规则进行扫描

本节将介绍如何使用测试自定义规则包来扫描项目。确保测试自定义规则包中仅包含希望测试的规则。在验证一个清晰的规则时，首先使用安全编码规则包扫描项目，然后使用自定义规则包和安全编码规则包再扫描一次。在使用某个函数且问题已被删除的上下文中验证规则是否与该函数匹配。

使用新的自定义规则扫描源代码：

1) 创建简单的源文件，其中的代码与其文件夹中的规则匹配。

2) 打开 Audit Workbench，单击 Advanced Scan(高级扫描)。屏幕上会显示 Browse for Root Directory(浏览根目录)窗口。

3) 选择包含源文件的文件夹，然后单击 OK(确定)。屏幕上会显示 Commandline Builder (命令行构建器)对话框。

4) 如果是 Java 项目，选择根文件夹，然后单击 Classpath Directory(类路径目录)。

5) 单击 Next(下一步)。Commandline Builder: Stages of Fortify SCA Analysis(命令行构建器：Fortify SCA Analysis 操作步骤)窗口会显示在屏幕上。

6) 单击 Manage Rulepacks(管理规则包)。

7) 如果未安装测试自定义规则包，单击 Add Custom Rulepack(添加自定义规则包)，然后安装规则包。

8) 清除所有安全编码规则包并自定义规则包(测试包除外)。

注意：在测试清晰的规则时，使用安全编码规则包和自定义规则包进行扫描。

9) 单击OK(确定)。

10) 单击 Run Scan(运行扫描)。

FPR 文件会显示在 Audit(审计)窗口中。

验证函数是否与规则匹配，Fortify SCA 将规则与函数一一匹配。利用这些说明来确保规则能够匹配所有希望匹配的函数，即便是规则类型可能从结果中删除了问题。

以下规则类型可能不会产生问题：

- Cleanse or validation rules：删除经由函数的数据中的感染标识；
- Source rules：为从该函数返回的数据添加一个或多个感染标识。这些规则可能/不可能产生问题；
- Sink rules：将带有特定感染标识的数据到达函数的情况识别为问题。

查看匹配函数的规则：

1) 在函数面板中，选择 Show All(显示全部)和 Group by function(按函数分组)。屏幕上会显示源代码中的所有函数。

2) 验证规则是否正确匹配了函数：

a) 右键单击函数，然后选择 Show Matched Rules(显示匹配的规则)。屏幕上会显示 Matched Rules(匹配的规则)窗口。它会显示匹配该函数的所有规则。

b) 定位测试中的自定义规则的规则 ID。

c) 单击 OK(确定)以关闭窗口。

d) 对规则所需匹配的每一个函数执行同样的操作。

3) 要查看函数的使用位置，右键单击函数，然后选择 Find Usages(查找使用)。

Search Results(搜索结果)选项卡会显示该函数的位置列表。其中包括文件的路径和函数使用的代码行数。

注意：这是验证规则匹配所需函数的最准确方式。

4) 单击函数的位置以在代码导航面板中显示该函数。

列出规则报告的各个问题：这部分说明介绍了如何查找和验证与特定规则相匹配的问题。

查找匹配某一规则的所有问题：

1) 在 Issue (问题)面板中，单击 All(全部)选项卡。

2) 在Group by (分组方式)中，选择New Issue(新问题)。屏幕上会显示新问题列表。

3) 选择所需问题，然后单击Details(详细信息)选项卡。

4) 滚动至末尾，然后复制规则 ID。

5) 在 Search(搜索)字段中，输入 ruleid: 然后立即粘贴规则 ID。例如： ruleid:97B9518A-F1BC-44CE-BEB1-D5FBDDFCCF9D

6) 按Enter键，屏幕上会显示所有与搜索标准相匹配的问题。

7) 验证规则是否报告了所需的问题的类型和数目。

列出从扫描中删除的问题：发现在扫描期间删除问题的规则并非易事，这部分说明介绍了如何显示所有在扫描期间删除的问题。要显示准备删除的项，需要首先使用安全编码规则包扫描代码，然后使用安全编码规则包和自定义的规则包(包含验证规则)扫描相同的代码库。

查找删除的问题：

1) 在Issue(问题)面板中，单击All(全部)选项卡。

2) 在 Group by(分组方式)中，选择 Removed Issues(删除问题)。

屏幕上会显示此扫描和最后一个扫描之间的问题列表。

自定义规则包的疑难解答：

扫描时报告的语法错误：如果以下错误发生在通过新规则运行扫描的过程中，在 XML View(XML 视图)中打开规则，并验证规则的每一个元素信息是否完整。错误表示规则的格式无效。

[1380] Rule "11F48876-58A8-4A48-8D78-C52E9295DC7E":语法错误。

检查扫描错误和警告：

1) 选择Tools(工具)，再选择Project Summary(项目汇总)。

2) 在Project Summary(项目汇总)面板中，单击Analysis Information(分析信息)选项卡，如图8-49所示。

3) 单击 Warnings(警告)选项卡。屏幕上会显示扫描警告和错误。

预期的问题未显示：自定义规则可能不匹配，原因如下：

- 函数或方法、类和包或命名空间的类型与函数在代码中的使用方式不匹配；
- 由源规则添加的 TAINT 标识与 sink 规则中的 TAINT 标识条件不匹配；
- TAINT 标识被 pass-through 规则意外删除；
- 匹配这个问题的可见性过滤器。选择 Options(选项)，然后选择 Show Hidden Issues(显示隐藏问题)。在 All(全部)选项卡中，选择 Group by <none>(任意分组)，右键单击该

问题，然后选择 Why is this issue hidden? (这个问题为何隐藏？)，Filters (过滤器)选项卡上会突出显示 Visibility Filter (可见性过滤器)。

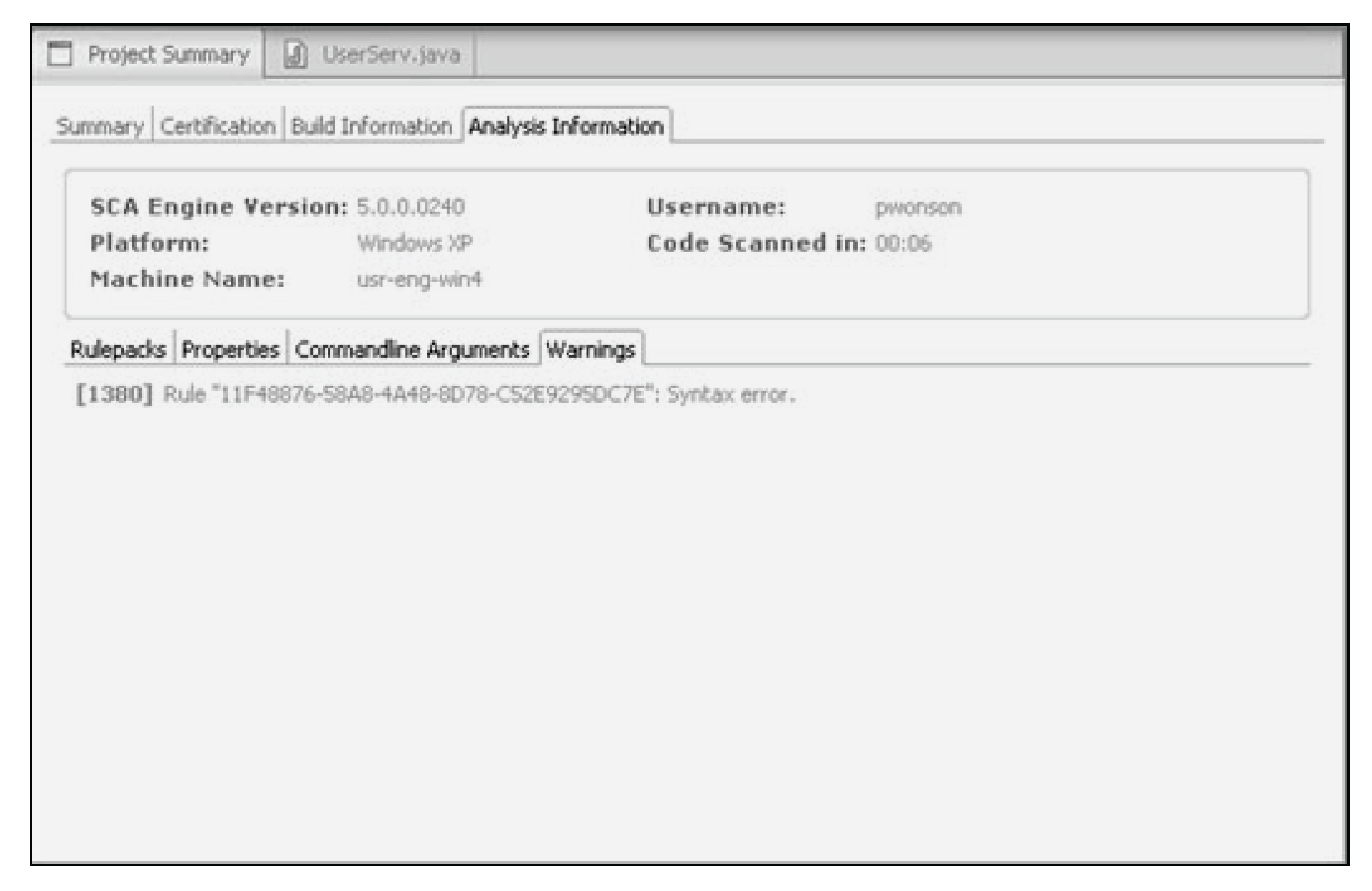



图 8-49　扫描信息

## 8.7.7　故障排除与支持

### 1. 故障排除

1) 利用日志文件进行故障排除

如果遇到运行时错误，请查看日志文件以获取相关信息。

有关 Fortify 的错误，请参见<fortify_data_directory>/awb/log/awb.log

有关 Eclipse 的错误，请参见<fortify_data_directory>/awb/.metadata/.log

其中：

在 Unix 平台上，<fortify_data_directory>是指目录~/.fortify

在 Windows 平台上，<fortify_data_directory>是指 Documents and Settings 下特定用户目录下的 Local Settings/Application Data/Fortify

2) org.eclipse.swt.SWTError 错误消息

在 Unix 系统上，Audit Workbench 可能无法启动，并会显示下列错误：

org.eclipse.swt.SWTError: No more handles [gtk_init_check() failed]

如果看到这个错误，则请确保 X11 的配置正确无误，而且已经设置了 DISPLAY 变量。

3) 内存不足错误

如果遇到内存不足的错误，而且使用的不是 Macintosh OS X 平台，可设置 AWB_VM_OPTS 环境变量来配置更大的堆大小。

例如，要为 Audit Workbench 分配 700 MB，则将该变量设置为 -Xmx700M。当指定该选项时，确保分配的内存未超过可用的物理内存，否则会降低效能。

作为一项指导原则，假设没有执行其他内存密集型进程，分配的内存不应超过系统可用内存的 2/3。

如果使用的是 Macintosh OS X，请编辑下列文件，将-Xmx500m 参数改为-Xmx700m 或更高：

<install_directory>/Auditworkbench.app/contents/MacOS/eclipse.ini

如果超过 JVM 在以下平台上强制执行的限制，则 Audit Workbench 可能无法加载，并显示错误：JVM terminated.Exit code=1。

- Linux 2.4 - 1800 MB
- Linux 2.6 - 2650 MB
- Windows 2000 - 1500 MB
- Windows 2003 - 1500 MB
- Windows XP - 1250 MB
- Mac OS X - 1800 MB
- AIX 5.2 - 无限制
- Solaris 8 - 1800 MB

4) 重置默认视图

如果曾经关闭或者移动过面板，如 Issues (问题)面板或 Summary (摘要)面板，则可以在 Options (选项)菜单中选择 Reset Default Views (重置默认视图)，即可将用户界面重置回原始的默认状态。

### 2. 报告错误和请求增强功能

用户反馈在产品的成功过程中必不可少。要请求增强功能、补丁，或者报告错误，请发送电子邮件到技术支持中心：

techsupport@fortify.com

请务必在电子邮件主体中包含下列信息：

产品：Audit Workbench

版本号：要确定版本号，请在 Audit Workbench 界面上选择 Help(帮助)，然后选择 About AuditWorkbench(关于 Audit Workbench)。

平台：(例如 PC)

操作系统：(例如 Windows 2000)

当请求增强功能时，请包含有关增强功能的描述。

当报告错误时，请提供足够的详细信息，以便最大限度地重现该问题。描述得越详细，技术支持部门就能越快地分析并修复该问题。同时要包含从发生该问题时开始记录的日志文件(或其中的相关部分)，以及对该问题严重程度的评价：

- 阻断性的
- 致命性的
- 重大的

- 一般的
- 次要的
- 轻微的

# 8.8　HP Fortify 实时安全分析器的主要特征

## 8.8.1　Fortify RTA 概述

Fortify RTA 是 Fortify 公司独有的在软件运行时进行安全防护的软件。它可以理解为“软的 IPS”，提供对 Web 应用系统运行时刻的防护和监控功能。RTA 的独特之处是它仅仅是一堆代码，通过静态插桩方式与 Web 应用系统的二进制代码结合后，就可以在 Web 应用系统内部工作。实时地了解、跟踪并分析 Web 应用系统的运行状况，当有恶意攻击数据进入系统时，它会及时地阻止攻击发生，为应用系统提供及时地防护；同时，它会把所有关于攻击的信息详细地记录下来，发送至控制台，从攻击的 When、What、Where、How 以及 Who 等方面，多维度报告攻击行为的信息，让运维人员或者安全管理人员及时地了解上线的应用系统在生产过程中遭受到的黑客攻击，以及应用系统自身所存在的安全漏洞等相关技术的详细数据和信息。它使得软件主动防御黑客成为可能。图 8-50 显示了工作原理图。

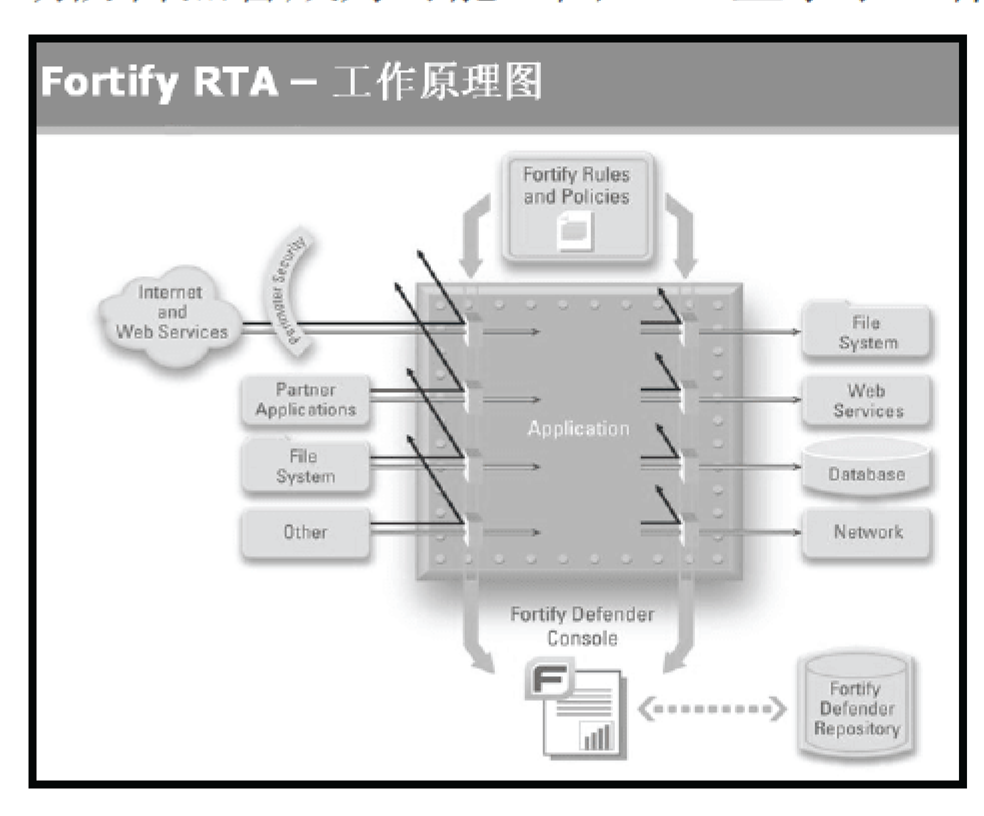



图 8-50　Fortify RTA 工作原理图

## 8.8.2　Fortify RTA 工作原理

“Fortify RTA，它通过对 Web 应用系统的每个 DNA 注入安全因子来增强应用系统自身的防攻击能力”，这是 OWASP 组织创始人 Mark Curphey 给出的形象化描述。正如 Mark Curphey 所说，Fortify RTA 是根据 AOP——“面向切面编程”的原理，通过对 Web 应用系统的可执行代码(不需要源代码)进行静态分析，找出所有的输入点(Input)和输出点(Output)，插入安全切面，即 Fortify RTA 的安全防御机制。因此 Fortify RTA 的安全检测机制就结合到了应用系统内部中，与应用系统的执行代码成为一体，就如同 RTA 是一剂安全防御疫苗被注入到了应用系统中。从应用系统内部形成防护网。这样一来，当结合了 RTA 的应用系统在生产环境上受到黑客攻击时，系统中的 RTA 就可以及时地对其进行防御了。

## 8.8.3　Fortify RTA 控制台

Fortify RTA 的功能之二就是可以把应用系统遭受的攻击的详细技术信息记录并报告出来，方便系统运维人员或安全管理人员及时地了解生产环境上的应用系统的安全状况。如图 8-51 所示，报告从 When、What、Where、How 以及 Who 等多维度地呈现，便于相关人员及时制订应对策略。

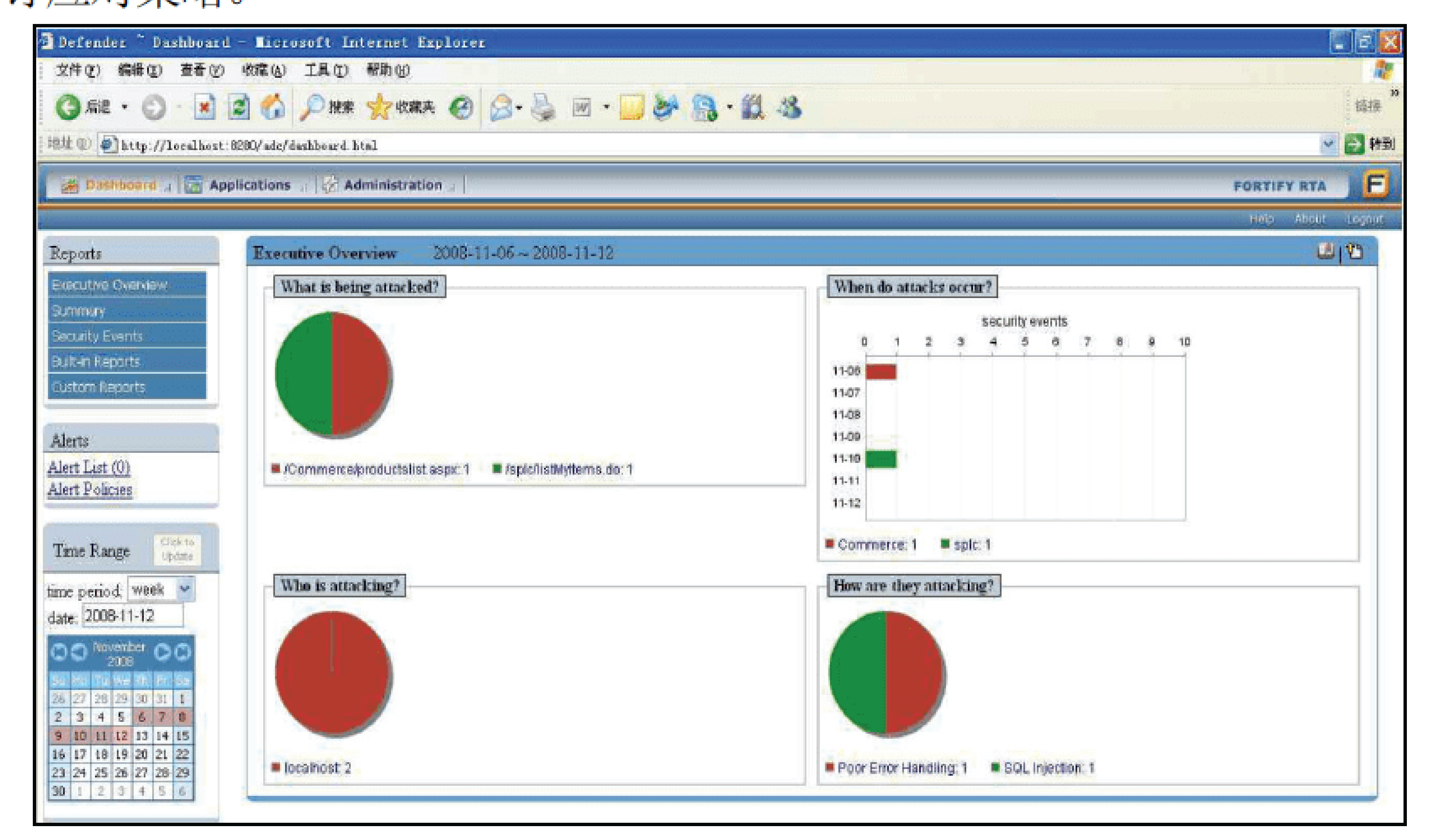



图 8-51　Fortify RTA 报告控制台

## 8.8.4　Fortify RTA 能够防御的攻击种类

Fortify RTA 能防御目前所有黑客常用的、危害较大十几种攻击方式，具体如下：

- Cross-SiteScripting/Cross-SiteScripting:PoorValidation
- HTTPResponseSplitting
- PoorErrorHandling:UnhandledException
- PrivacyViolationGeneric
- SQLInjection/SQLInjection:Hibernate/NHibernate
- SessionFixation
- PrivacyViolation:CreditCard/SocialSecurityNumber
- CreditCardFraud
- CrossSiteRequestForgery
- DecoyTampering
- ForcefulBrowsing
- LinkSpam(Reflected&Persistent)
- Probing/Probing:BrokenLink/CommandInjection/Worm
- SystemInformationLeakage

## 8.8.5　Fortify RTA 应用的平台

语言： J2SE 1.4 或更新版本； J2EE 1.3.1 或更新版本； .NET 1.1 和 2.0

操作系统：Windows 2K, 2003, XP； Linux RH9, ES 3.0, Fedora； Mac OS X； Solaris 8 和 9

应用服务器：Tomcat 4.X,5.X, 6.X；Weblogic 8.x,9.X；Jboss, Websphere 6.0, IIS

Web 浏览器：Linux 和 Solaris Mozilla 1.7 或更新版本；Firefox 1.0 或更新版本；Windows IE 6, Firefox 1.0 或更新版本；Macintosh Firefox 1.0 或更新版本

# 8.9　HP Fortify 程序跟踪分析器(PTA)

## 8.9.1　Fortify PTA 概述

Fortify PTA 是 Fortify Software 公司最新推出的软件安全动态测试工具。它可以帮助软件功能测试人员(QA 人员)在不需要任何黑客渗透测试技术的情况下，对 B/S 应用系统进行动态的安全性测试。Fortify PTA 利用动态污染传播方法(DYNAMIC TAINT PROPAGATION)对测试人员输入的测试数据进行跟踪，分析数据在程序中的传递和执行情况，从中分析程序中潜在的安全漏洞，帮助功能测试人员报告安全测试结果，整个过程自动完成，功能测试人员只需做常规的功能测试即可。解决了缺乏安全渗透测试人员和功能测试人员缺乏黑客知识的难题。

## 8.9.2　Fortify PTA 工作原理

Fortify PTA 利用动态污染传播方法的特点，首先对软件的二进制代码进行插桩分析，找出所有 Source、Sink 代码，建立检测机制。然后对程序进行常规的功能测试。测试过程不需要测试人员输入任何带攻击性的测试数据，Fortify PTA 会根据功能测试自动找出软件中所有可能因外部输入数据而造成的安全问题，并根据漏洞清晰地报告出来。如图 8-52 所示。

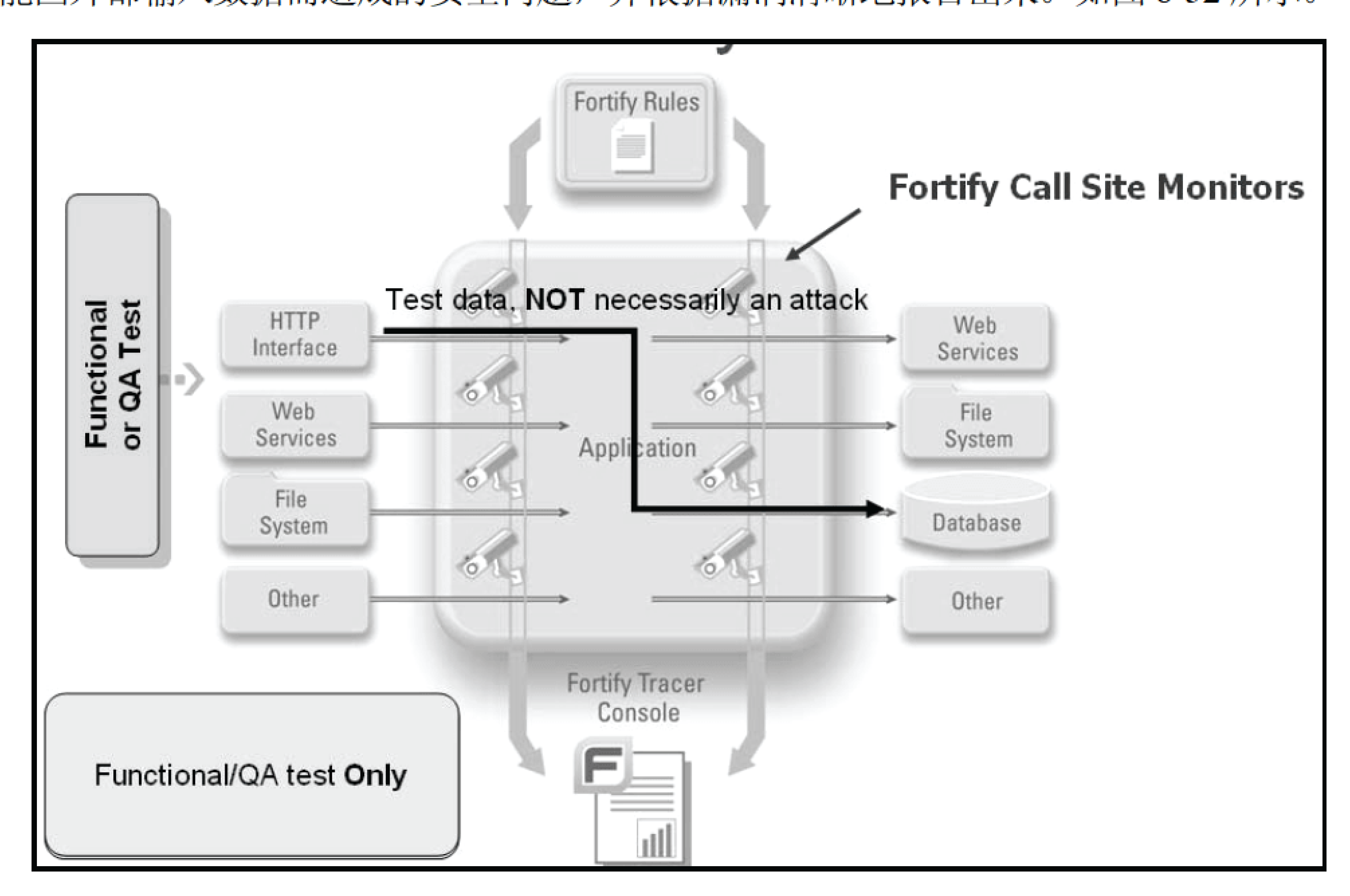



图 8-52　Fortify PTA 工作原理图

## 8.9.3　Fortify PTA 特点

总结Fortify PTA的主要特点有：

1) 不需要特殊的攻击性测试数据，让 QA 人员都可以做安全测试。解决测试缺乏安全知识、攻击知识的难题。

2) 由于直接跟踪外部输入数据，所以能够很真实、有效地找出系统中最严重、最关键的安全问题。

3) 与功能并行，速度快，效率高，很容易与现有测试流程结合。同时利用功能测试覆盖率高的优点，大大提高了动态安全测试的覆盖率。

4) 如图 8-53 所示，Fortify PTA 根据漏洞类别清晰地报告出来，单击报出的每一个漏洞，Fortify PTA 可以给出该漏洞在什么地方产生的，它的 Source 和 Sink 分别在哪里等清晰而又详细的报告。

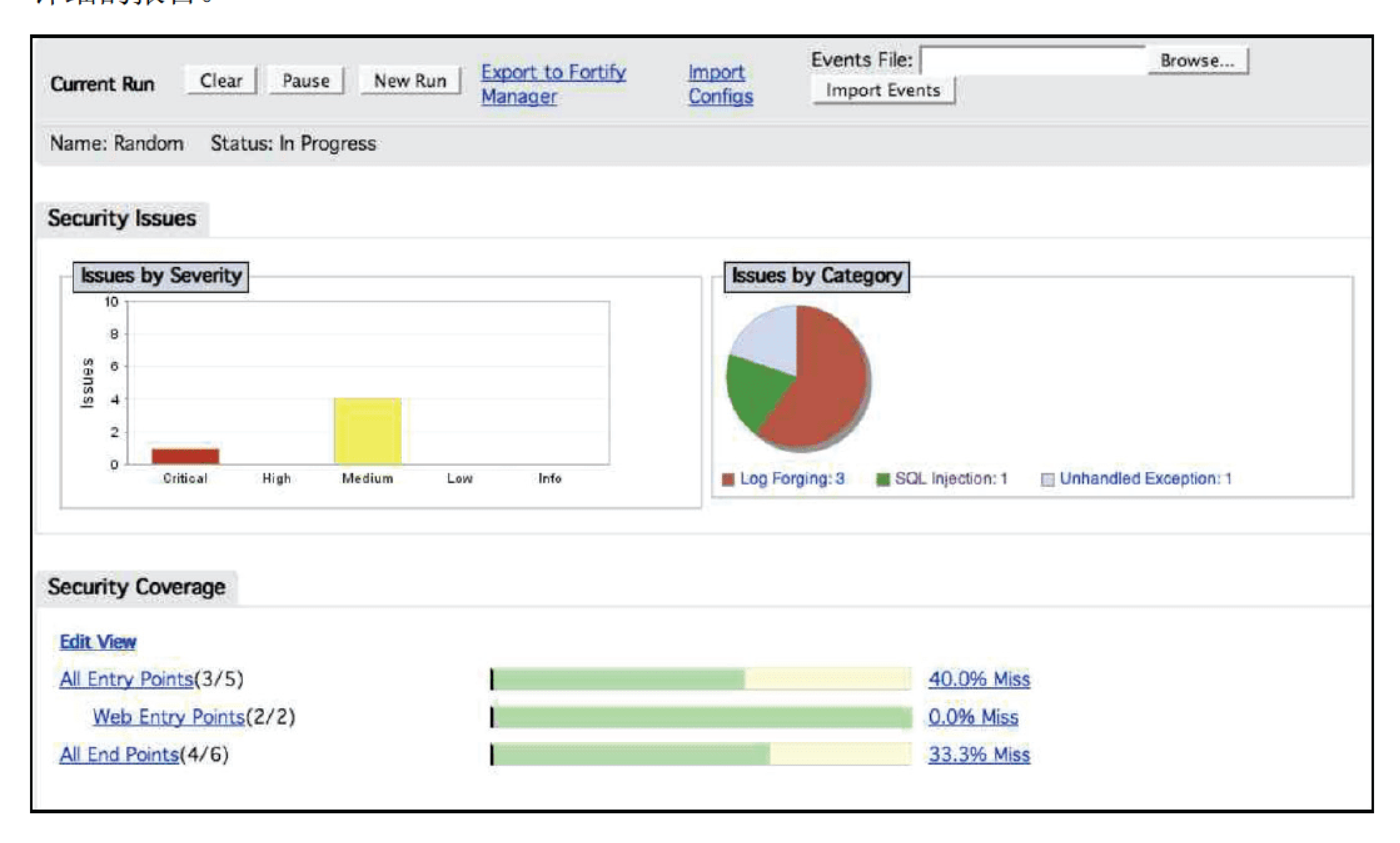



图 8-53　Fortify PTA 安全测试报告

## 8.9.4　Fortify PTA 可以分析和防御的安全漏洞种类

- SQL Injection
- Cross-Site Scripting: Reflective
- Cross-Site Scripting: Persistent
- Command Injection
- Arbitrary URL Redirection
- Path Manipulation
- HTTP Response Splitting
- Unhandled Exception
- Privacy Violation: Social Security Number

- Privacy Violation: Credit Card Number
- Log Forging
- Xpath Injection
- LDAP Injection
- Private data being stolen
- Security leaks
- Fraudulent behavior
- Web site defacement
- Phishing attacks
- Escalation of privileges

### 8.9.5　Fortify PTA 应用的平台

语言：J2SE 1.4 或更新版本；J2EE 1.3.1 或更新版本；.NET 1.1 和 2.0

操作系统：Windows 2K, 2003, XP；Linux RH9, ES 3.0,Fedora；Mac OS XSolaris 8 和 9

应用服务器 Tomcat 4.X, 5.X, 6.X；Weblogic 8.x,9.X；Jboss, Websphere 6.0, IIS

Web 浏览器：Linux 和 Solaris Mozilla 1.7 或更新版本；Firefox 1.0 或更新版本；Windows IE 6, Firefox 1.0 或更新版本；Macintosh Firefox 1.0 或更新版本

## 8.10　本章小结

本章介绍了 HP Fortify 的三大分析器，其中主要讲述了静态代码分析器(Fortify SCA)，包括主要特征、不同系统下的安装特点、分析原理、分析过程、代码扫描方式、转换源代码和 Audit Workbench 工具的使用，深入讨论了静态代码分析器 5 大分析引擎，介绍安全漏洞扫描机制并对扫描结果进行分析。以上所述也充分体现了 SCA 代码分析和对扫描结果进行处理的强大功能。

另外，本章还对实时安全分析器(RTA)和程序跟踪分析器(PTA)进行了简要介绍，包括工作原理、控制台情况、防御攻击种类以及各自应用的平台。可通过了解两个分析器的实际用途和发展状况，来保障企业或个人信息的安全。

# 参考文献

[1] Brian Chess, Jacob West. 安全编程—代码静态分析[M]. 董启雄等译. 北京：机械工业出版社，2008.3.
[2] Brian Chess, Jacob West. Secure Programming with Static Analysis[M]. Boston: Addison-Wesley Professional, 2007.9.
[3] Michael Sikorski, Andrew Honig. 恶意代码分析实践[M]. 诸葛建伟, 姜辉, 张光凯译. 北京：电子工业出版社，2014.4.
[4] 李匀. 网络渗透测试—保护网络安全的技术、工具和过程[M]. 北京：电子工业出版社，2007.12.
[5] Paco Hope, Ben Waltller. Web 安全测试[M]. 付鑫等译. 北京：清华大学出版社，2010.3.
[6] David Kennedy, Jim O'Gorman, Devon Kearns, Mati Aharoni.Metasploit.渗透测试指南[M]. 诸葛建伟等译. 北京：电子工业出版社，2012.1.
[7] Patrick Engebretson. 渗透测试实践指南：必知必会的工具与方法[M]. 缪纶，只莹莹，蔡金栋译. 北京：机械工业出版社，2012.11.
[8] 王文君， 李建蒙. Web 应用安全威胁与防治——基于 OWASP Top 10 与 ESAPI[M]. 北京：电子工业出版社，2013.1.
[9] Dafydd Stuttard. 黑客攻防技术宝典: Web 实战篇[M]. 石华耀译. 北京：人民邮电出版社，2009.8.
[10] 钟晨鸣，徐少培. Web 前端黑客技术揭秘[M]. 北京：电子工业出版社，2013.1.
[11] 吴翰清. 白帽子讲 Web 安全(纪念版)[M]. 北京：电子工业出版社，2014.6.
[12] Matt Bishop. Computer Security；Art and Science[M]. Pearson Education，Ltd.，2004.1.
[13] 何泾沙. 信息安全导论[M]. 北京：机械工业出版社，2012.2.
[14] Steve Suehring, Robert L. Ziegler. Linux 防火墙[M]. 何泾沙等译. 北京：机械工业出版社，2006.6.
[15] 任伟. 软件安全[M]. 北京：国防工业出版社，2010.7.
[16] 郭克华. 软件安全实现-安全编程技术[M]. 北京：清华大学出版社，2010.6.
[17] Dieter Gollmann. 计算机安全学[M]. 张小松等译. 北京：机械工业出版社，2008.4.
[18] Dieter Gollmann. Computer Security[M]. New Delhi, India: Wiley India, 2007.1.
[19] Michael Howard, Steve Lipner. 软件安全开发生命周期[M]. 李兆星，原浩，张钺译. 北京：电子工业出版社，2008.1.
[20] Michael Howard, Steve Lipner. The Security Development Lifecycle: SDL[M]. Microsoft Press, 2006.6.

[21] John Viega Gary Mcgraw. 安全软件开发之道—构筑软件安全的本质方法[M]. 殷丽华等译. 北京：机械工业出版社，2014.3.
[22] David Rice. 软件安全败局启示录[M]. 赵俐等译. 北京：机械工业出版社，2009.4.
[23] 王清. ODAY 安全：软件漏洞分析技术[M]. 北京：电子工业出版社，2008.4.
[24] Ian Sommerville. 软件工程[M]. 程成译. 北京：机械工业出版社，2011.5.
[25] Hewlett-Packard Development Company, L.P. HP WebInspect User Guide[M]. 2014.4.
[26] Hewlett-Packard Development Company, L.P. HP Fortify Audit Workbench User Guide[M]. 2013.9.
[27] Hewlett-Packard Development Company, L.P. HP Fortify Static Code Analyzer Install and Configuration Guide[M]. 2013.9.
[28] Hewlett-Packard Development Company, L.P. HP Fortify Static Code Analyzer User Guide[M]. 2013.9.